Optimum Test Strategy for SoCs

Including wrapper and TAM design and optimization

by

Zahra sadat Ebadi

B.A.Sc., Sharif University of Technology, 2000

A THESIS SUBMITTED IN PARTIAL FULFILMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF APPLIED SCIENCE

in

The Faculty of Graduate Studies

(Department of Electrical and Computer Engineering)

We accept this thesis as conforming
to the required standard

THE UNIVERSITY OF BRITISH COLUMBIA

December 4, 2002

Abstract

Using the large number of transistors available on a chip, designers have already managed to put an entire system on a single chip. These are referred to as System-on-a-Chip (SoC). Being able to rapidly develop, manufacture, test, debug and verify complex SoCs is crucial for the continued success of the electronics industry.

In the problem of SoC test integration three issues need to be addressed: wrapper design, TAM design and test scheduling. In this thesis, a novel wrapper design method is introduced to minimize the core test time, the number of test I/O pins and the required ATE memory. While previous methods for wrapper design only minimize the test time, the proposed method considers all of these factors in the test cost.

Also a novel TAM based on time domain multiplexing (TDM-TAM) is introduced. This TAM is P1500 compatible and uses a P1500 wrapper. Its characteristics are flexibility, scalability, and reconfigurability.

Contents

List of Tables

List of Figures

Preface

These papers have been published earlier:

1. Z. Ebadi, A. Ivanov, "Design of an optimal test access architecture using genetic algorithm", Proc. IEEE Asian Test Symposium (ATS), pp. 205-210, 2001.

2. Z. Ebadi, A. Ivanov, "Design of an Optimal Test Access Architecture under Power and Place-and-Route Constraints using GA", Proc. IEEE Latin-American Test Workshop(LATW), pp. 154-159, 2002.

3. Z. Ebadi, A. Ivanov, "Time Domain Multiplexed TAM: Implementation and Comparison", will apear in Proc. Design Automation and Test in Europe, March 2003.

Acknowledgements

I would like to dedicate this thesis to my parents, Seyyed Mohammad and Seyyedeh Hajar Ebadi, and my dear husband, Alireza Nasiri Avanaki, who was my mentor during my studies in UBC.

I had the pleasure of being a member of the SoC research group at UBC. I would like to thank my fellow researchers in this group for their support and ideas.

This work is supported by Micronet, the Canadian Microelectronics Corporation, and the Natural Sciences and Engineering Research Council of Canada. Their support is greatly appreciated.

I would like to thank Dr. Andre Ivanov for his encouragement and invaluable advice. Also I would like to thank Dr. Resve Saleh for his helpful discussions.

Chapter 1

Introduction

1.1 Motivation

Spurred by technology which makes millions of gates per chip now available, system-level integration is evolving as a new paradigm, allowing an entire system to be built on a single chip, referred to as *System-on-a-Chip* (SoC; Figure 1.1). The advantages of an SoC over its conventional multi-chip counterpart include higher performance, lower power consumption, higher density and lower weight.

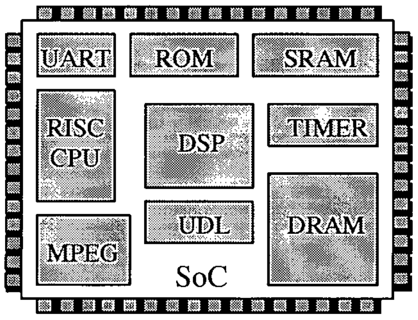


Figure 1.1: Example of System-on-Chip (SoC).

In SoC design, using complex, pre-designed and pre-verified functional blocks (modules) allows greater on-chip functionality, and leads to shorter product development cycles. These reusable modules are called *Embedded Cores*, while the reuse design style is known as a core-based design. A few examples of reusable cores include CPUs, DSPs and embedded memories.

Being able to rapidly develop, manufacture, test, debug and verify complex SoCs is crucial for the continued success of the electronics industry. To make production practical and cost effective, however, the International Technology Roadmap for Semiconductors (ITRS) identifies a number of major hurdles to be overcome [5]. Among these problems, testing and diagnosis of SoCs is the most important. Many experts believe that testing SoC chips will be the bottleneck of future designs if issues of DFT (design for testability) for SoCs are not addressed [41, 54]).

ITRS introduces "Manufacturing Test Cost" as a difficult challenge (through 2007) to be solved in the short term. The most important issues involved in manufacturing test costs are shown in Table 1.1.

Table 1.1 makes it obvious that DFT methods that reduce test cost by minimizing test I/O pin, test time, and equipment reuse, are desired. Also, since using such DFT methods leads to longer test development time, automatic test program generators are required. The research reported in this thesis focuses on DFT methods for SoCs to reduce test costs and also develop tools to automate generating test circuitry for SoCs.

Table 1.1: Manufacturing test cost issues (from Table 19 in [5]).

Difficult Challenge through 2007	**Summary of Issues**
Manufacturing Test Cost	Test cell throughput enhancements are needed to reduce manufacturing costs. Device test needs must be managed through DFT to enable low-cost manufacturing test solutions; including reduced pin count test, equipment reuse, and reduced test time. Automatic test program generators are needed to reduce test development time.

1.2 Research Goals

For the problem of SoC test integration three issues need to be addressed: wrapper design, test access mechanism (TAM) design and test scheduling. The wrapper forms the interface between the embedded core and its system chip environment, and provides switching capability between the normal operation mode and test mode. A TAM is used to deliver test data to the cores and also to transfer test response from the cores to the sink where it is evaluated. Test scheduling focuses on determining the start time of the various core tests, such that no resource conflict occurs with respect to the test access infrastructure from I/O pins to core terminals, and vice versa [21].

The test cost of an SoC depends on three major factors: (1) test time, (2) test input/output (I/O) pins, and (3) Automatic Test Equipment (ATE) machine resources. In an SoC integration problem, it is desirable to minimize test cost by minimizing the test cost factors.

Therefore, the goal here is to design and develop a computer-aided-design (CAD) tool which facilitates shorter test development time. Using this tool SoC designers can design the "optimal" test circuitry (with the minimum test cost).

The first two issues concerning SoC testing and their optimization are addressed extensively in this thesis, both theoretically and experimentally. Software has also been developed for optimal SoC testing in two steps: a core wrapper design for each individual core within the SoC, and a TAM design for the entire SoC.

In the first step, the wrapper that optimizes the core's test cost (by minimizing all the important test cost factors), is designed for each core. In the second step, the optimal TAM configuration for SoC are derived to optimize test cost (by minimizing test time and test I/O pins).

1.3 Thesis Organization

The remainder of this thesis is organized as follows. Chapter 2 provides an overview of basic concepts of SoC testing such as wrapper, TAM, and so forth. It also describes previous work on wrapper and TAM design and optimization. This work on TAM optimization [19, 20] is addressed in this chapter.

Chapter 3 describes wrapper design and optimization. In this chapter, a novel wrapper design method is introduced to minimize the core test time, the TAM width (translates to test I/O pins) and the required ATE memory. Superior performance of my method as compared to existing methods is proved through several experiments.

Chapter 4 proposes a novel TAM based on time domain multiplexing (TDM-TAM), which can especially handle multi-frequency SoCs. It also provides TDM-TAM characteristic details and a comparison to the two other approaches.

The thesis is concluded with a summary, possible directions for future research, and the contributions this project has made to the SoC test research community.

Chapter 2

Background and Previous Works

This chapter presents an overview of basic concepts of SoC testing, along with a background on wrapper and TAM design and optimization methods. Also, the focus and contribution of this research project are presented.

2.1 Overview of SoC Testing

In a core-based design, the system integrator is responsible for putting together a test strategy (for all embedded cores, and User Defined Logics (UDLs) and interconnect wiring) for the entire system chip.

The SoC test integration steps are shown in Figure 2.1. The wrapper concept and existing wrappers in the literature are discussed in Section 2.1.1. The definition of TAM and different types of TAM design are reviewed in Section 2.1.2. For test scheduling, given a set of tests for each core in the SoC and a set of test resources (e.g., TAM), a test plan for conducting the tests on the SoC is derived. Test scheduling is the last step of SoC test integration, however, it is not part of test circuitry design and can usually be done after tape out; thus, it is not in the scope of this research.

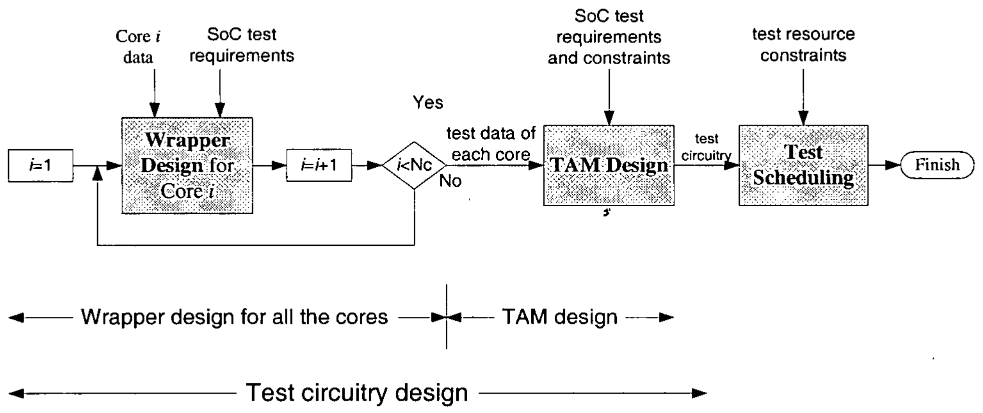


Figure 2.1: SoC test integration steps.

2.1.1 Wrapper

A *wrapper* is a shell around the core that integrates it with its surroundings and provides an isolation mechanism for testing purposes. A wrapper acts like a switch between normal functional access and test access, providing a mechanism for core test data access and core test isolation; it allows the tester to control core inputs and observe core outputs via the TAM, so that the core's internal test can be re-applied at the SoC level. To avoid any damage to the SoC, a core has to be isolated when it, or one of its neighboring cores or UDLs, is tested.

Furthermore, a wrapper should have additional modes so that the isolated core can interface with available test access paths and assume internal states necessary for the core's own test execution, as well as testing external interconnects and UDLs. Therefore, a well-designed wrapper can handle both core-internal testing and core-external testing. In order to achieve these objectives, the following capabilities are potentially required at core terminals:

- **Input Observation**: This capability allows logic values of the core input terminals (which are supplied by external logic) to be observed by the wrapper.

- **Input Control**: This wrapper function allows test data to be applied to the core input terminals.

- **Input Constraint**: Allows forcing or limiting core input terminals to fixed logic values. Input constraint can be useful during testing to prevent damage to the core, reduce power consumption, and so forth.

- **Output Observation**: This capability allows for the observation of logic values of the core output terminals (which are propagated from internal logic) at the wrapper.

- **Output Control**: This wrapper function allows test data to be applied at the core output terminals by the wrapper, such that the test data can be propagated to the system chip logic external to the core.

- **Output Disable**: Allows forcing tri-sate core outputs to their inactive state. This is useful in preventing damage to other tri-state drivers on the same bus during the test application.

- **Output Constraint**: This allows constraining appropriate non-tri-sate core output to fixed logic values in order to prevent damage to the logic external to the core and reduce its power consumption.

There are several proposed core test wrappers in the literature. A core test wrapper called *TestShell* [41] has been proposed by Marinissen, and is currently being used by Philips. This wrapper supports four basic modes: (1) normal (functional), (2) core test, (3) interconnect test and, (4) bypass. In this approach, TAMs are called TestRail. In principle, a TestShell is connected to the same TestRail at both input and output. Therefore, the TAM input plug and the TAM output plug of a TestShell normally have the same width.

Varma and Bhatia of Duet Technologies described a very similar wrapper, called *Test Collar* [50]. Aside from the different naming of basically similar features, the main difference between this wrapper and TestShell is that the Test Collar does not have a bypass feature.

The IEEE P1500 Standard for an Embedded Core Test (SECT; under development) [4] consists of two components: a *Core Test Language* to facilitate the test knowledge from the core provider to the core user, and a *Core*

Test Wrapper [44]. The P1500 wrapper (see Figure 2.2), which satisfies all of the functional requirements mentioned above, is composed of the following:

- *Wrapper Boundary Cells*: These are associated with the core terminals. They provide controllability as well as observability. Together, these cells make the *Wrapper Boundary Register.*
- *Wrapper Instruction Register* (WIR): This is used to load instructions to control the operation of the wrapper.
- *Bypass Register*: This is the bypass for the Serial Interface Layer (SIL).
- *Wrapper Interface Port* (WIP): This is for the control of wrapper registers via SIL. It comprises control and data signals for accessing DFT features of the target core.

A detailed description of P1500 core test wrapper elements is reported in Appendix A.

The P1500 wrapper connects to one mandatory one-bit wide TAM, Wrapper Serial Input/Output (WSI/WSO), and zero or more scalable-width TAMs (TAM-in/TAM-out). A minimal compliant implementation has only the single-bit TAM plug, through which both test control values for the WIR, as well as test stimuli and responses are transported. Envisaged typical usage has one multi-bit TAM next to the mandatory TAM. In this case, access to the bulk test data is performed along the multi-bit TAM, while the single-

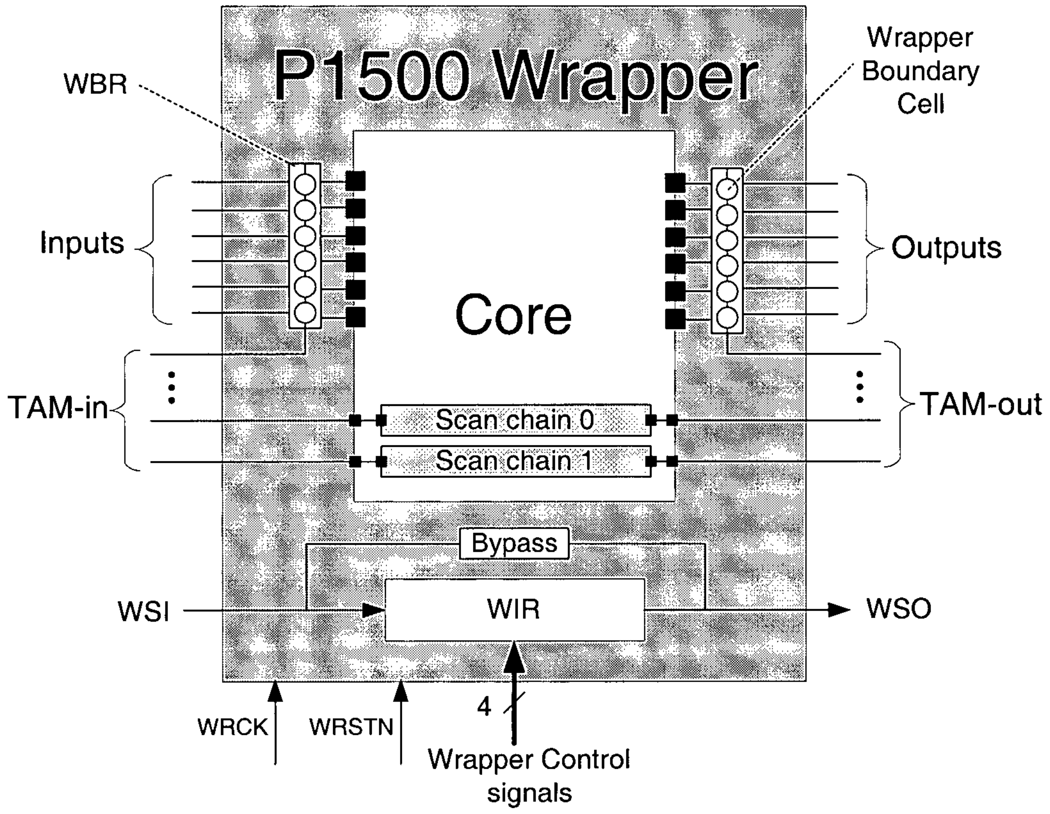


Figure 2.2: Conceptual view of IEEE P1500 wrapper [44].

bit TAM is used to program the WIR, and possibly transport test data in a silicon debug scenario. TAM-in and TAM-out need not have the same width.

This research on optimal wrapper design (Chapter 3) is based on the P1500 wrapper standard. The previous works on wrapper design and optimization are reported in Section 2.2.

2.1.2 TAM

Since cores in an SoC are not directly accessible via chip inputs and outputs, special access mechanisms are required to test them at the system level.

Zorian et al. [54] proposed a generic conceptual test access architecture for embedded cores, with the following components: source, sink, and TAM (Figure 2.3). A TAM is used to deliver test stimuli from the source (which generates test stimuli) to cores, and also to deliver responses from cores to the sink (which evaluates test responses).

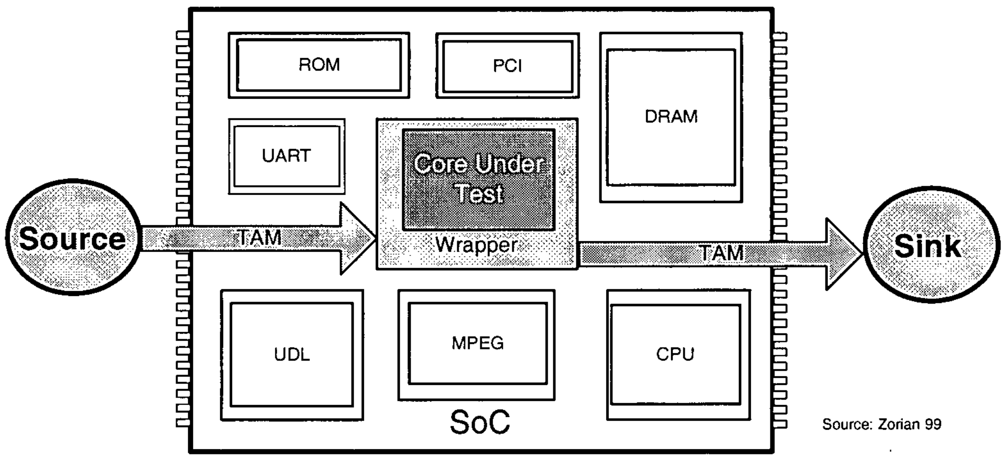


Figure 2.3: Conceptual TAM architecture.

In this architecture (Figure 2.3), both source and sink can be either on-chip or off-chip. The TAM is not only the physical mechanism that connects source and sink to the core, but also includes the control signals needed for this connection.

The IEEE P1500 proponents, accepting the fact that test schemes for each core cannot be standardized, specifically decided not to standardize the design for the TAM. Hence, the TAM design is left to the SoC integrator.

Several TAM architectures are suggested. These architectures can be classified into four categories: (1) multiplexing, (2) serial connection, (3) indirect access and (4) bus-based connection.

In the first category, multiplexing is uses to access the cores. The simplest method in this category directly multiplexes the test pins to the primary inputs/outputs (I/O) [30]. Another method modifies the cores such that each core has a transparent mode for testing [22]. There are several problems with the multiplexing TAM methods, such as limited scope of use for future complex SoCs, large overhead area, long test time, and non-scalability of the architecture.

TAMs in the serial connection category [40, 49, 52] use the established IEEE 1149.1 standard. For a few cores on an SoC, it may be possible to spend time transporting the test vectors serially to the cores. However, as the number and complexity of the cores increases, a serial solution based on the IEEE 1149.1 standard, or its variants, proves too costly in terms of test time.

There are some proposed methods that implement TAM without a direct path from I/Os to each core. One of these methods is Networked Indirect and Modular Architecture (NIMA) proposed in [45], where the emphasis is placed on modularity, generality, and configurability of the architecture. The basis of NIMA is the establishment of indirect digital communication paths among cores using packet-switching connections. It is assumed that all test-related communications destined and/or originated to or from the SoC need to be communicated by the on-chip network.

Bus-based connection schemes are the most common TAM architecture. A number of different variations of the scheme are reported in [41, 50, 53]. The idea is to have parallel access to the cores using a shared medium on which data is broadcasted. Bus access control is usually provided by tri-state switches (Figure 2.4). In terms of trading-off increased overhead area to reduce test access time, bus-based architectures are the most efficient TAM schemes suggested to date.

Marinissen and others [41] suggest a topology (as a bus-based TAM) where cores are connected in a rail configuration, and buses have different widths, fan-in, and fan-out. If needed, each core can be bypassed to access the next one in line, and control is achieved by a serial connection. This architecture is conceptually illustrated in Figure 2.5 (the control signals for the bypass elements are not shown). Test data is normally passed through the bypass elements and the corresponding cores are inhibited in this mode. When the data has to be applied to a core, the bypass element is inhibited

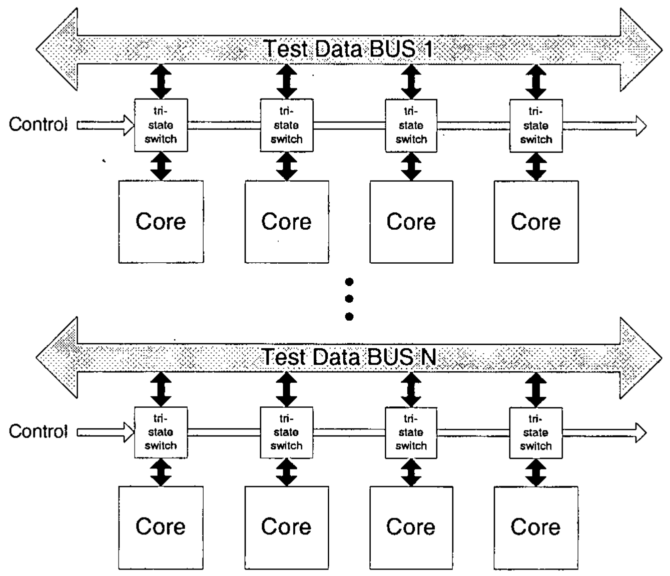


Figure 2.4: TAM using bus-based connection.

and the corresponding core reads the data. The wrapper configuration used for each core (not shown in the diagram) closely resembles that of the first draft of IEEE P1500.

Using the bus width as the varying parameter, TAM architectures based on a bus provide some degree of adaptability and configurability for the designer: increased overhead area is traded-off for reduced test access time to design for the best case. In Section 2.3 previous work on TAM optimization are reported.

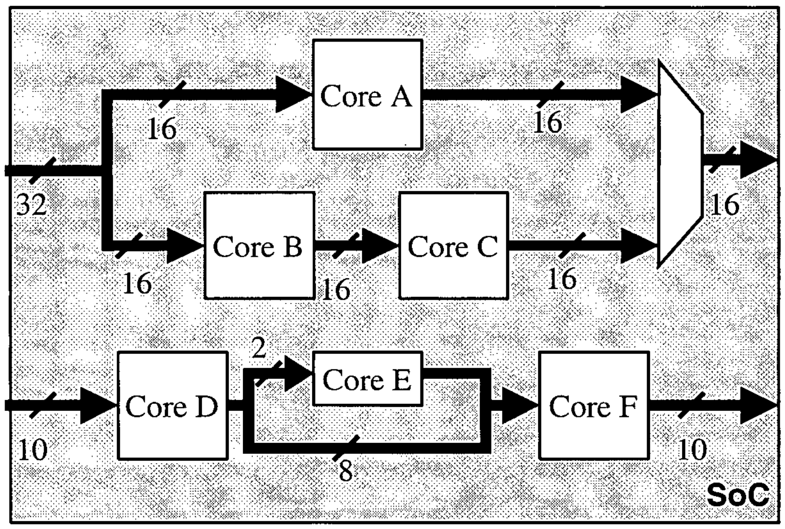


Figure 2.5: TESTRAIL architecture for TAM [41].

2.2 Wrapper Optimization Methods

The existing wrapper optimization methods are mostly based on the P1500 wrapper standard. The publications on P1500, however, provide only general guidelines for wrapper design, and details are left to the core user. For example, a wrapper may provide width adaption in case of a mismatch between the core I/O width and TAM width (e.g. by serial-parallel and parallel-serial conversion). Interconnection between the wrapper cells, core-internal scan chains and TAM plugs effect core test time, required TAM width, and also required ATE memory.

Marinissen [42] proved that the unbalanced TAM scan chains (i.e. the scan chains formed by internal scan chains and the core's inputs/outputs having unequal lengths; see Section 3.2.1) of the wrapper lead to longer core test time. He suggested a method to balance the TAM scan chains of the wrapper, aimed at minimizing core test time [42].

Iyengar [32, 34, 35] not only improved Marinissen's method of reaching shorter test times, but also considered minimizing TAM width while keeping the test time at the minimum.

Gonicari [24] addressed useless memory allocation (UMA) as an issue in wrapper design. To reduce on-chip control when feeding embedded core's multiple scan chains, the test vectors are augmented with useless data to account for their unequal lengths (Section 3.4). Gonicari proposed a new test methodology which merges core wrapper design and ATE memory management problems. His core wrapper design is capable of finding the minimum number of TAM scan chain partitions, so that for each partition, useless memory is minimized.

2.3 TAM Optimization Methods

An efficient TAM should reduce test costs by minimizing test application time and test I/O pins. Since there is no standard for a TAM configuration, each type of TAM has its own optimization method. Usually for any proposed TAM, optimization is also addressed as part of the TAM solution. However, sometimes the problem of TAM optimization for a general bus-based configuration is also addressed.

Chakrabarty [14, 16, 31] addressed several issues in optimal TAM design with respect to test time, for example, assignment of cores to the test buses, the distribution of the test width among multiple test buses, and the estimation of TAM width required to satisfy an upper bound on testing time. All of these problems are proved to be NP-complete, and therefore, in [14, 16] integer linear programming (ILP) is used to solve them.

As an extension of previous work, Chakrabarty [15] proposed a method of designing the optimum TAM (using ILP) considering system level constraints on power consumption and place-and-route (arising from the functional interconnections amongst the cores). All of his work [14–16] required serialization at all the cores' I/Os.

As part of this research, the TAM optimization problem was. The same issues as Chakrabarty (cores assigned to buses, width distribution and required width estimation) were investigated. However, the necessity of serialization constraint was removed, which allowed the system to handle serial or parallel loading of test data for any core; this improves the test time for up to 40%. The implementation of the proposed method was built around a genetic algorithm (GA). In Figure 2.6 this method [19] is compared to Chakrabarty's [14].

In another work, the proposed method have been improved to handle the

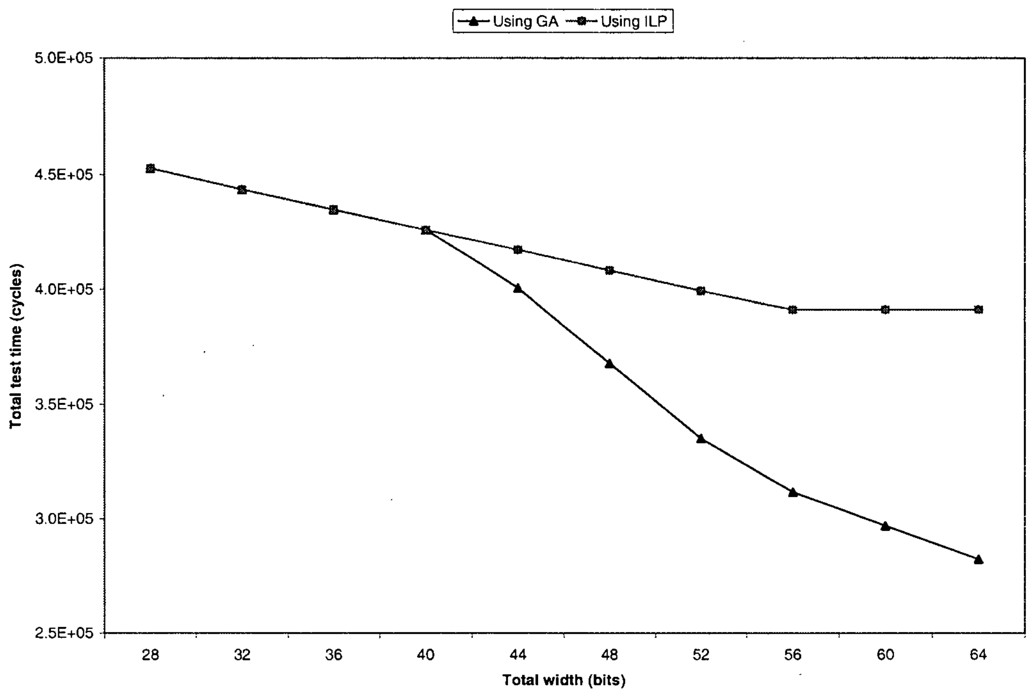


Figure 2.6: Comparison of result using ILP [14] and GA [19].

power and place-and-route constraints [20]. In this work the proposed system was able to find an optimal test architecture with four options: (1) without any constraint, (2) with a power consumption constraint, (3) with a place-and-route constraint, and (4) with power consumption and place-and-route constraints. An example of the results of the proposed system is shown in Figure 2.7.

In some works the problem of wrapper and TAM optimization are combined and solved together [33–36]. For example, in [33], the wrapper design algorithm (which minimizes test time and TAM width) calculates test time

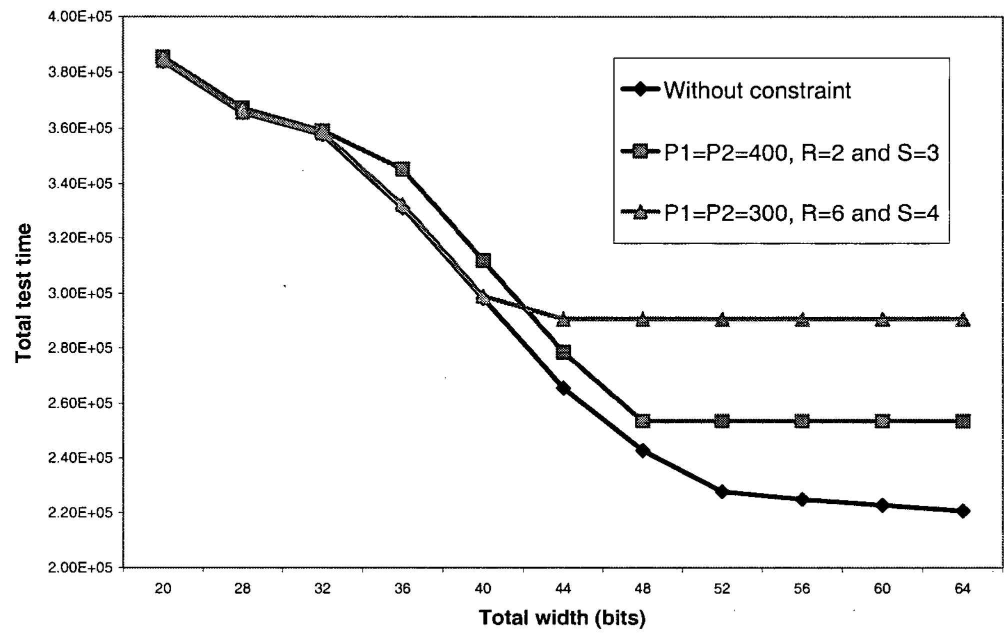


Figure 2.7: Example of result of the system in [19].

values that are used by a mathematical model for TAM optimization.

2.4 Focus and Contributions

The goal for this research project is to build a tool for designing a test circuitry for SoC test integration. As it is shown in Figure 2.1, the first step is wrapper design. In this thesis, a new method for wrapper design regarding all important test cost factors, including test time, required width, and required ATE memory is proposed. As reported in Section 2.2, all previous works consider only one or two of the major factors. The proposed approach not only considers all the major test cost issues, but also, its performance is better than the previous methods (e.g., lower test time as compared to the methods that just minimize test time [32, 42]). Also, the proposed method is implemented as a tool for optimal wrapper design for a given core. The details of wrapper design and optimization are discussed in Chapter 3.

In Chapter 4, a novel TAM based on time domain multiplexing (TDM-TAM) is proposed. This TAM is P1500 compatible, and uses a P1500 wrapper. TDM-TAM characteristics are flexibility, scalability, and reconfigurability. Also, this TAM could be very useful for testing multi-frequency SoCs because with TDM, we can manipulate frequency. The concept of TDM-TAM is expanded theoretically and experimentally. The optimization method is tailored specifically for TDM-TAM based on my previous work in TAM optimization [19, 20]. The efficiency of TDM-TAM is proved by a comparison with two other TAMs.

The contributions of this thesis are summarized as follows:

1. A novel wrapper design algorithm considering all major components of test cost.

2. A novel TAM based on time domain multiplexing with the following special characteristics:

    - P1500 standard compatible

- Reconfigurability (even after fabrication)
- Low overhead area and acceptable test time
- Ability to handle multi-frequency SoCs without expensive ATE and complicated ATE programming

3. Specific optimization algorithms for TDM-TAM based on previous work on TAM optimization is reported in [19, 20].

Chapter 3

Wrapper Design

This chapter introduces the important issues in wrapper design. Next, the TAM chain design problem is addressed and existing methods for this problem are reviewed. A new method is introduced that improves wrapper design to minimize area and ATE (Automatic Test Equipment) resources, as well as core test time. Finally, wrapper design for flexible cores is elaborated.

3.1 Wrapper Design Issues

There are many important factors to consider in the design of wrappers. Anything that has an impact on the test cost can be a factor. The most important part of the test cost is known to be determined by the test time. The silicon area is also a factor of the test cost. Another factor is the number of I/O pins in the test cost, because the number of I/O pins is limited by area. That is, as the number of I/O pins increase, we need a larger area so the test cost increases. The resources of an ATE machine can also be counted as a factor of the test cost. Since we need more resources from an

ATE , such as memory, we have to employ a more expensive ATE, which has a large impact on the test cost. To summarize, the factors that should be considered are as follows:

- **Test Time**
    - *Core Internal Test Time*: core test time
    - *Core External Test Time*: interconnection test time
- **Area**
    - *Silicon Area*
    - *I/O Pins*
- **ATE Resources**
    - *Memory*
    - *I/O Pins*

3.2 Wrapper Design w.r.t. Test Time

3.2.1 TAM Chain Design

One of the most important tasks in wrapper design, which has a direct impact on the core test time, is to make the interconnection between the wrapper boundary cells, the core internal scan chains, and the TAM lines (plugs). We refer to this task as *TAM chain design.* Also, we call the elements that make up a TAM chain (wrapper input cells, core internal scan chains and wrapper output cells), *TAM chain elements.* Access to the core for testing is already guaranteed if all the TAM chain elements are accessible from the TAM plugs. In this section, it will be shown that the ordering and partitioning of TAM chain elements has a large impact on the size of the resulting test vectors set. Also, it is obvious that the size of the test vector set is an important cost factor in the testing of ICs. When the test vector set is large, not only the application test time is long, but also, expensive pieces of equipment with large pin memories to store those vectors are required to test the IC. Therefore, the reduction of the size of the test vector results in the reduction of test cost.

Wrappers are used for both core-internal and core-external tests. Designing the wrapper to optimize the test vector set for a core-internal test might lead to a conflict with the test vector set optimizing for a core-external test. Usually, the core-internal circuitry is much larger than the circuitry used to interconnect the cores. Therefore, the test data volume involved in core-internal testing is much larger than the test data volume for core-external testing. Also, in many cases, the wrapper is designed by the core provider to whom the circuit environment in which the core is used is unknown, and data about the core-external test is not available at the wrapper design time.

Therefore, we give priority to optimizing the core-internal test vector set, and the wrapper is designed without considering the core-external test.

The test time T (in clock cycles) of a core is determined [23] by the scan-in length s_i, the scan-out length s_o, and the number of test patterns N_P:

$$T = (1 + \max(s_i, s_o)) \cdot N_P + \min(s_i, s_o) \tag{3.1}$$

This equation assumes that the scan-out operation of one pattern is overlapped with the scan-in of the subsequent test pattern, and that the actual test application costs one clock cycle. In regular scan testing, typically $s_i = s_o$. Since in the wrapped cores, the scan chains also include wrapper cells for core terminals, and the number of core input terminals might differ from the number of core output terminals [42], then s_i and s_o might be different for these. Note that this formula is valid even for non-scan-testable cores, for which $s_i = s_o = 0$.

Theorem 3.2.1 *For a core with N_I inputs, N_O outputs, N_S scan chains of length $l_1, l_2, \cdots, l_{N_S}$ respectively and N_P test patterns, the lower bound on the core test time, T, is given by the following:*

$$T_{lower} = N_P l_{max} + N_P + l_{max}$$

and the minimum TAM width required to achieve this test time's lower bound *is given by W_{min}, as follows:*

$$W_{min} = \lceil \frac{L + \max(N_I, N_O)}{l_{max}} \rceil^*$$

where $l_{max} = \max_{i=1,\dots,N_S} l_i$ and $L = \sum_{i=1}^{N_S} l_i$.

Proof Since in best case, each TAM chain element is on one TAM line, so s_i and s_o cannot be less than l_{max}. In the best case the longest internal scan chain is on a private TAM chain making $s_i = s_o = l_{max}$. Substitution of these values into Equation (3.1) gives the lower bound for the test time $N_P l_{max} + N_P + l_{max}$. Also, the minimum width required to achieve this lower bound is in the case where $s_i = s_o = l_{max}$ for all the TAM chains. As well, we know that $s_i \geq \frac{L+N_I}{W}$, and $s_o \geq \frac{L+N_O}{W}$, giving us the following:

$$s_i \geq \frac{L+N_I}{W}, s_i = l_{max} \Longrightarrow l_{max} \geq \frac{L+N_I}{W} \Longrightarrow W \geq \frac{L+N_I}{l_{max}}$$

$$s_o \geq \frac{L+N_O}{W}, s_o = l_{max} \Longrightarrow l_{max} \geq \frac{L+N_O}{W} \Longrightarrow W \geq \frac{L+N_O}{l_{max}}$$

Therefore, $W \geq \frac{L+\max(N_I,N_O)}{l_{max}}$. However, W can only take integer values, so $W \geq \lceil \frac{L+\max(N_I,N_O)}{l_{max}} \rceil$. ∎

Theorem 3.2.2 *For a core with N_I inputs, N_O outputs, N_S scan chains of length $l_1, l_2, \cdots, l_{N_S}$ respectively, and N_P test patterns, the upper bound on*

*$\lceil . \rceil$ *denotes rounding to the closest larger integer.*

the core test time, T, is given by the following:

$$T_{upper} = (1 + L + \max(N_I, N_O))N_P + L + \min(N_I, N_O)$$

where $L = \sum_{i=1}^{N_S} l_i$.

Proof The worst case for test time is when only a one-bit wide line is used for TAM. In this case all the scan chains and inputs and outputs are on one TAM line, so $s_i = L + N_I$ and $s_o = L + N_O$. By substituting in Equation (3.1), the upper bound of the test time is $(1+L+\max(N_I, N_O))N_P + L + \min(N_I, N_O)$. ∎

In our test architecture design approach, we distinguish two types of cores:

- Cores with *fixed-length* scan chains: cores for which the numbers and the lengths of the internal scan chains are fixed. Examples of such cores are hard cores and firm cores after scan insertion. When designing an SoC-level test architecture, we have to cope with the fixed scan chain parameters.

- Cores with *flexible-length* scan chains: cores for which the numbers and lengths of the internal scan chains are determined during test architecture design. Examples of such cores are soft cores and firm cores before scan insertion. Such cores provide better opportunities to minimize test time, as we can tune their scan chain parameters to fit the overall SoC.

In all the sections of this chapter, except Section 3.5, we assumed all the cores have fixed-length scan chains.

3.2.2 Ordering of TAM Chain Elements

From the set of all TAM chain elements, two non-disjoint subsets are involved in the loading and unloading of test patterns. The wrapper input cells and the core-internal scan chains (we refer to these as the *input elements*) participate in loading of test patterns, while the wrapper output cells and core-internal scan chains (we refer to these as the *output elements*) participate in the unloading of test patterns. In order to reduce s_i and s_O, it is best to order the items in any TAM chain such that the *input elements* are at the head, and the *output elements* are at the tail of the TAM chain. However, considering that core-internal scan chains are in both sets, the elements should be ordered in the following way: (1) wrapper input cells, (2) core-internal scan chains, and (3) wrapper output cells.

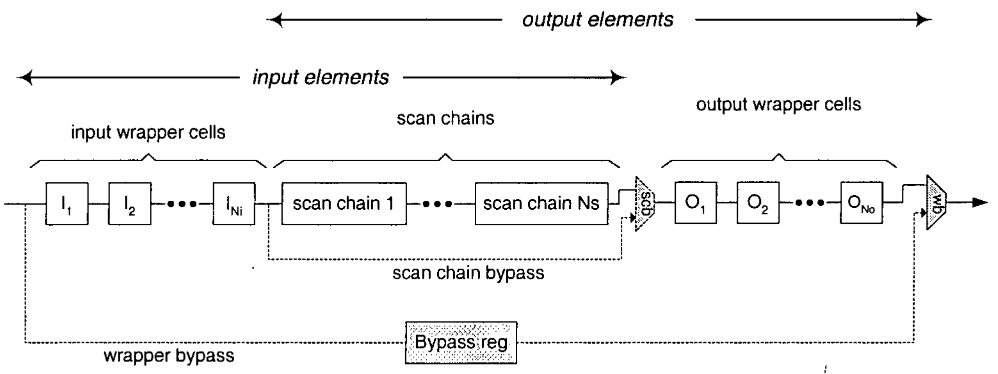


Figure 3.1: Ordering of TAM chain elements (optional items are dashed).

In Figure 3.1, the generic template for a single TAM chain is shown. As another option, we can provide a bypass for the core-internal scan chains, and at the cost of a multiplexer and an additional control wire, we can reduce the length of the access chain by bypassing them during the core-external test. We can also provide a bypass for the entire TAM chain in this wrapper. Such a bypass is practically useful when multiple cores are concatenated into a single TAM, such as the case in the Daisy-chain architecture described in [8]. Cores which are not being tested can be bypassed in order to reduce the access length to cores which are being tested. As multiple cores are daisy chained into one TAM, this might lead to long TAM wires, and hence to long propagation delays.

3.2.3 Partitioning of TAM Chain Elements

Usually the TAM width is limited because of limited IC pins, silicon area, and so forth. Therefore, in many cases, the total number of TAM elements is much larger than the width of the TAM (the number of TAM lines). In these cases, it is required to partition the set of TAM elements into a number of subsets equal to the number of available TAM chains.

The partitioning of TAM elements over TAM wires has a direct impact on scan-in time, s_i, and scan-out time s_o. Partitioning determines core test time; hence, we look for a partitioning of the TAM elements which yields to the minimum test time.

For cores with fixed-length scan chains, the TAM element partitioning is challenging. Suppose that we have the core and the following information: N_I functional inputs, N_O functional outputs and N_S scan chains with lengths $l_1, l_2, \ldots, l_{N_S}$. We want to design a wrapper for this core with TAM width (TAM plugs) W to minimize the core test time.

The partitioning problem can be formulated as finding an assignment of all TAM elements to one of the available TAM chains such that the core test time, Equation 3.1, is minimized. This problem can be formalized as follows [42]:

Problem 1 Partitioning of TAM Chain Elements (PTE)

Assume a set of W identical TAM chains and a set $\mathcal{WI} = \{I_1, I_2, \cdots, I_{N_I}\}$ of wrapper input cells, where each wrapper input cell has a length 1, $l(I_j) = 1$. Given a set of $\mathcal{S} = \{s_1, s_2, \cdots, s_{N_S}\}$ of core-internal scan chains, where scan chain s_j has length l_j, $l(s_j) = l_j$. Given a set of $\mathcal{WO} = \{O_1, O_2, \cdots, O_{N_O}\}$ of wrapper output cells with length of 1, $l(O_j) = 1$. We define, the group of TAM elements as: $\mathcal{TE} = \mathcal{WI} \cup \mathcal{S} \cup \mathcal{WO}$, with $|\mathcal{TE}| = N_I + N_S + N_O$. A TAM partition $\mathcal{P} = \{P_1, P_2, \cdots, P_W\}$ of $\mathcal{TE}$ is such that it divides $\mathcal{TE}$ into W disjoint sets, one for each TAM chain. For each TAM chain j, $1 \leq j \leq W$ we define a set of input elements $IN_j = P_j - \mathcal{WO}$ and a set of output elements $OUT_j = P_j - \mathcal{WI}$. Thus, the scan-in length of TAM chain j, becomes $l(s_{i_j}) = l(IN_j)$, which is actually the length of the wrapper input cells (or the number of wrapper input cells) and the sum of the lengths of the scan chains in TAM i. Also, the scan-out length of TAM chain j becomes $l(s_{o_j}) = l(OUT_j)$. Hence, the scan-in and scan-out length for the whole core with this partitioning becomes $s_i(\mathcal{P}) = \max_{j=1}^{j=W}(s_{i_j})$ and $s_o(\mathcal{P}) = \max_{j=1}^{j=W}(s_{o_j})$, respectively. Thus, the core test time for this partitioning would be $\mathcal{T}(\mathcal{P}) = \{1 + \max(s_i(\mathcal{P}), s_o(\mathcal{P}))\} \cdot N_P + \min(s_i(\mathcal{P}), s_o(\mathcal{P}))$. From the latter definitions, the PTE problem can be stated as follows:

Find an optimal TAM partition $\mathcal{P}^*$, which satisfies $\mathcal{T}(\mathcal{P}^*) \leq \mathcal{T}(\mathcal{P})$, for all partitions $\mathcal{P}$ of TE.

To solve the PTE problem, we use a three-step approach.

1. First, we assign the core-internal scan chains in $\mathcal{S}$ to W TAM chains, such that the maximum sum of scan lengths assigned to a TAM chain is minimized. The resulting partition is named $\mathcal{P}_S$.

2. Second, we assign the wrapper input cells in $\mathcal{WI}$ to TAM chains on top of $\mathcal{P}_S$, such that the maximum scan-in time of all TAM chains is minimized.

3. Third, we assign the wrapper output cells in $\mathcal{WO}$ to TAM chains on top of $\mathcal{P}_S$, such that the maximum scan-out time of all TAM chains in minimized.

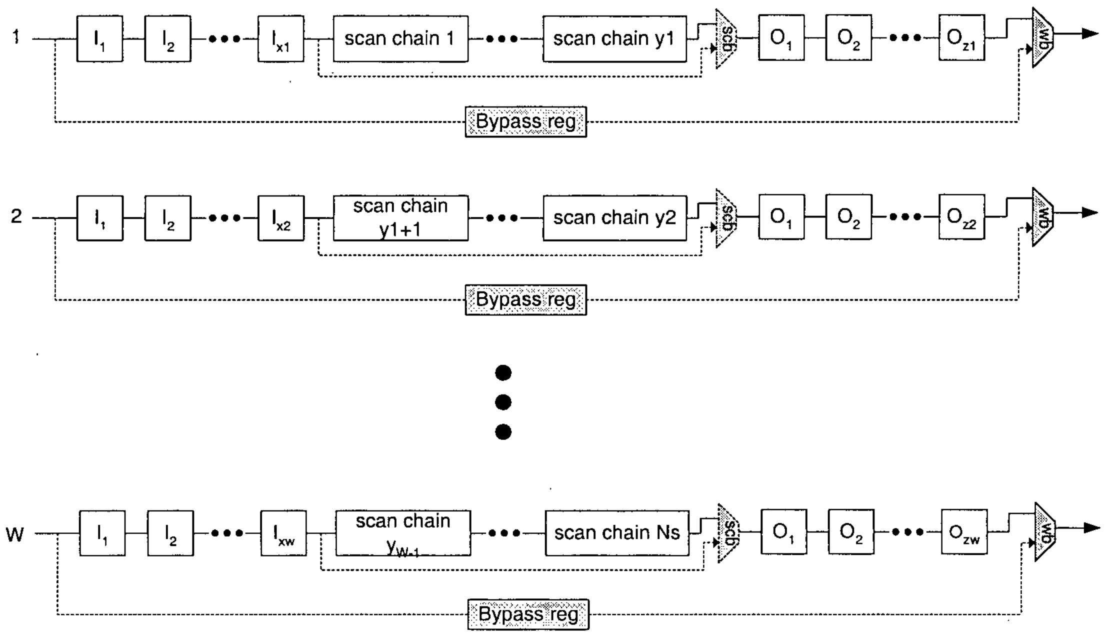


Figure 3.2: A conceptual view of TAM elements partitioning over TAM chains.

Note that wrapper input and output cells have lengths of one. Therefore, Steps 2 and 3 can yield an optimal solution in linear processing time, once Step 1 is completed. The first step is to solving the problem of partitioning scan chains over TAM chains, which can be formulated as follows.

Problem 2 Partitioning of Scan Chains (PSC)

Assume a set of $\mathcal{S} = \{s_1, s_2, \cdots, s_{N_S}\}$ of core-internal scan chains, where scan chain s_i has length l_i, i.e. $l(s_i) = l_i$, and a set of W identical TAM chains. A *scan partition* is a partition $\mathcal{PS} = \{Ps_1, Ps_2, \cdots, Ps_W\}$ of $\mathcal{S}$ into W disjoint sets, one for each TAM chain. TAM chain i, $1 \leq i \leq W$, contains all scan chains in Ps_i. The scan length of scan partition $\mathcal{P}$ is given $s(\mathcal{P}) = \max_{i=1}^{i=W}(l(Ps_i))$, where for any $\mathcal{X} \subset \mathcal{S}$, $l(\mathcal{X}) = \sum_{s\in\mathcal{X}}(l(s))$. Find an optimal scan partition $\mathcal{P}^*$, which satisfies $s(\mathcal{P}^*) \leq s(\mathcal{P})$, for all partitions $\mathcal{P}$ of $\mathcal{S}$ into W subset.

Theorem 3.2.3 *The PSC problem is NP hard.*

Proof The PSC problem [39] is equivalent to the well known problem of *minimum makespan* problem, described in Appendix B. To show that PSC is NP hard, we consider the decision problem version of PSC: given N_S scan chains with length $\{l_1, l_2, \cdots, l_{N_S}\}$ and W TAM chains, is there any partition (of N_S scan chains into W subset, P_i) such that, $\sum_{j\in P_i}(l_j) \leq \mathcal{K}$, $\forall P_i$? This problem is exactly equivalent to the multiprocessor scheduling version minimum makespan problem. The multiprocessor scheduling (MPS) problem is stated as follows [47]:

Instance: There is a finite set A of "tasks", a "process time" $t(a)$ for each $a \in A$, a number $m > 0$ of "processors", and a "deadline", $D > 0$.

Question: Is there a schedule of A into m disjoint subsets, i.e. $A = A_1 \cup A_2 \cup \cdots \cup A_m$, such that $\max\{\sum_{a\in A} t(a)\,;\ 1 \leq i \leq m\} \leq D$?

This problem is known to be strongly NP-complete[47].

The equivalence between a decision version of PSC and a minimum makespan problem can be easily established by noting the correspondence between processors (m) and TAM chains (W), and between tasks and scan chains. The

Table 3.1: List Decreasing Algorithm.

```
Algorithm 1 [LD]
(assume $W \leq N_S$)
Sort $\mathcal{S}$ such that $l(s_1) \geq l(s_2) \geq \ldots \geq l(s_{N_S})$;
for $i := 1$ to $W$
    $P_i = s_i$;
for $i := W + 1$ to $N_S$
    select $k \in \{j | l(P_j) = \min_{1 \leq x \leq W} l(P_x)\}$;
    $P_k := P_k \cup \{S_i\}$;
return $\max_{1 \leq x \leq W} l(P_x)$;
```

deadline D corresponds to $\mathcal{K}$. Therefore, the decision version of PSC problem is NP-complete. Also, when a decision version of a combinatorial optimization problem is proven to belong to the class of NP-complete problems, then the optimization version is NP-hard [6]. So PSC problem is NP-hard. ∎

In the literature, various polynomial-time algorithms have been proposed for MPS [9, 18, 28, 47] that achieve near-optimal schedules. These methods are reviewed in the following.

Graham [25] proposed the *List Decreasing* (LD) algorithm (See Table 3.1), that first sorts the tasks in order of decreasing processing time. Then it assigns the task at the top to the minimally loaded processor. LD has a time complexity of $\mathcal{O}(N_S \log N_S + N_S \log W)$, in which W is the number of the processors (the width of TAM) and N_S is the number of tasks (scan chains). In Appendix B it is proved that the worst case performance ratio is $\frac{4}{3} - \frac{1}{3W}$.

Bin-packing and MPS problems are dual problems[27]. In the MPS problem, the goal is to pack several objects of given sizes into a given number of bins, and to minimize the maximum necessary capacity of the bins. The original formulation of MPS was scheduling jobs of different lengths to run on a given number of machines so that the whole system finishes as early as possible (these formulations are equivalent). In the bin-packing problem, the number of bins is variable, but their capacities are constant. The two problems are the duals of each other: both are known to be strongly NP-hard. Suppose we have a super-optimal solution to the bin packing problem, an instance that is known to use at most as many bins as the optimal, but may "overhang" a bit, by a factor of ε, making it infeasible. This solution then can be used to approximate the corresponding makespan problem in Table 3.2

Table 3.2: Using Bin-packing solution to solve MPS.

```
Let $L = \max\{\max_j s_j, \frac{1}{m}\sum_i s_i\}$.
Guess a median $d \in [L, 2L]$.
Find an $\varepsilon$-dual approximation for packing $W$ bins of size $d$.
If less than $W$ bins were used, search the smaller half of the interval.
If $W$ bins proved to be too few, search the larger half of the interval.
Repeat until length of the interval is less than 1.
```

The algorithm shown in Table 3.2 is very general. For example there is a method (Table 3.4) in which the *First Fit Decreasing* (FFD) algorithm is used for bin-packing [11, 38, 46]. Also, for upper bounds of the interval, the

List Decreasing algorithm is used.

Table 3.3: Using First Fit Decreasing to solve MPS.

```
Algorithm 2 (Using First Fit Decreasing)
Sort $\mathcal{S}$ such that $l(s_1) \geq l(s_2) \geq \ldots \geq l(s_{N_S})$
$MinCost := \frac{\sum_{i=1}^{N_S} l_{s_i}}{W}$;
$C_U := LD$;
$C_L = \lfloor \max(\frac{LD}{\frac{4}{3} - \frac{1}{3W}},\ l_1,\ MinCost) \rfloor$;
$d := C_L$;
while $d \leq C_U \wedge FFD(d) > W$ do
    $d := d + 1$;
return $\mathcal{P}$;

First Fit Decreasing : FFD(d)
Assume initially $P_j = \emptyset$ for all $j$
Sort $\mathcal{S}$ such that $l(s_1) \geq l(s_2) \geq \ldots \geq l(s_{N_S})$
for $i := 1$ to $N_S$
    $j := 1$
    while $l(P_j) + l_i > d$ do
        $j := j + 1$
    $P_j := P_j \cup S_i$;
return $\max\{j \mid P_j \neq \emptyset\}$;
```

As we mentioned above, MPS and bin-packing are equivalent, so from a standard algorithm providing a bin-packing solution we can extrapolate to solve MPS. There are some standard algorithms for bin-packing, such as FFD, Last Fit Decreasing, Worst Fit Decreasing, and Best Fit Decreasing (BFD) [11, 38, 46]. BFD is the best algorithm for our purposes because it looks for the best possible assignment (of scan chain to TAM chains) with the minimum cost (minimum makespan). Algorithm shown in Table 3.4 uses the BFD to solve PSC.

Example Consider core 6, the largest logic core from p93791 from the

Table 3.4: Best Fit Decreasing Algorithm (BFD).

```
Algorithm 3 (BFD)
Assume initially $P_j = \emptyset$ for all $j$
Sort $\mathcal{S}$ such that $l(s_1) \geq l(s_2) \geq \ldots \geq l(s_{N_S})$
for $i := 1$ to $N_S$
    find $P_{max}$ with current maximum length
    find $P_{min}$ with current minimum length
    assign $s_i$ to TAM chain $P$, such that $|l(P_{max}) - \{l(P) + l(s_i)\}|$ is minimum
    if there is no such TAM chain $(P)$ then assign $s_i$ to $P_{min}$
return $\mathcal{P}$;
```

ITC'02 SoC Benchmarks [7]. Core 6 has 417 functional inputs, 324 functional outputs, 72 bidirectional I/Os, and 46 internal scan chains. Seven scan chains have length of 500 bits, thirty have length of 520 bits, and nine scan chains have length of 521 bits. LD, the one that used FFD to solve MPS (Table 3.3) and BFD have been used to assign core internal scan chains to the TAM chains. The result is shown in Table 3.5.

From Table 3.5, the three algorithms, LD and BFD perform the same, but Algorithm 2 performs better. Here, LD and BFD perfomarance is the same baceause in both algorithm all the scan chains are sorted first, so best assignmnet (BFD) is the same as the assignmnet to the TAM chain with minimum length (LD). Algorithm 2 performance is better, for example, for $W = 9$, both LD and BFD output has a cost of 3081, but the Algorithm 2 solution has a cost of (makespan) 3000, which means saving 81 cycles for each pattern, considerable for a large number of test patterns.

The same result (better performance of Algorithm 2) has been derived

Table 3.5: The maximum scan-in/scan-out length of the assignment of core internal scan chains to TAM chains for core 6 of p93791 of ITC'02 benchmarks.

	makespan: max(l(P))		
TAM Width	Algorithm 1 (LD)	Algorithm 2 (using bin-packing)	Algorithm 3 (BFD)
1	23789	23789	23789
2	11904	11904	11904
3	8263	8263	8263
4	6202	6202	6202
5	5142	5142	5142
6	4141	4140	4141
7	3621	3620	3621
8	3101	3100	3101
9	3081	3000	3081
10	2581	2580	2581
11	2561	2560	2561
12	2080	2080	2080
13	2061	2060	2061
14	2060	2060	2060
15	2041	2000	2041
16-19	1560	1560	1560
20-21	1540	1540	1540
22	1521	1500	1521
23	1041	1041	1041
24-38	1040	1040	1040
39-42	1020	1020	1020
43-45	1000	1000	1000
46-∞	521	521	521

Table 3.6: Adding wrapper input/output cells to TAM chains.

Algorithm 4: Adding wrapper input/output cells to TAM chains

```
$i := N_I$ or $N_O$;
While $i > 0$, do
    find $P_{max}$ with current maximum scan-in/scan-out length
    find $P_{min}$ with current minimum scan-in/scan-out length
    if $l(P_{max}) < l(P_{min})$ then
        Add $l(P_{max}) - l(P_{min})$ wrapper input/output Cells to $P_{min}$
        i:=i-($l(P_{max}) - l(P_{min})$)
    if $l(P_{max}) = l(P_{min})$ then
        Add $\lfloor \frac{i}{W} \rfloor$ to all $P$s;
        if $i\%w \neq 0$ then
            Add 1 wrapper input/output cell to $i\%w$ $P$s;
return $\mathcal{P}$ (which we call it after that $\mathcal{P}_S$;
```

for differenet cores. Therefore, Algorithm 2, the one using a bin-packing solution, is chosen for the first step of wrapper design, the assignment of core internal scan chains to the TAM chains.

The second step is adding the wrapper input cells to the TAM chains, and the third step is to add wrapper output cells. These two steps are equal and both can be solved optimally in linear processing time. Algorithm 4 [†] in Table 3.6 is proposed for steps 2 and 3.

The result of wrapper design for core 6 of p93791 is shown in Figure 3.3. For this wrapper design we used Algorithm 2 to assign core internal scan chains to the TAM chains, and Algorithm 4 to add wrapper input/output cells. In Figure 3.3, the max$\{s_i, s_o\}$ vs. TAM width is plotted (for this core since $N_I > N_O$, so max$\{s_i, s_o\} = s_i$). We observe that as W increases, the

[†]In Algorithm 4 $i\%w$ is the remainder of i divided by w.

scan-in length decreases in a series of distinct steps. This is because as W increases, the core internal scan chains are distributed among a larger number of TAM chains. Thus the scan-in length decreases only when the increases in W are sufficient to remove an internal scan chain from the longest TAM chain. For example, when the internal scan chains of core 6 are distributed among 24 TAM chains, s_i is 1040 bits long. The value of s_i remains at 1040 until W reaches 39. We will use this trend to save in test I/O pins (for example, we can reach the same test time using 24 TAM lines instead of 38 TAM lines, saving 14 TAM lines for just one core). Another observation is that beyond a TAM width of 47, the test time does not decrease. This observation corresponds to Theorem 3.2.1: the lower bound on the test time based on the theorem is this (for this example $l_{max} = 521$ and number of test patterns $N_P = 218$):

$$T \geq l_{max} \cdot N_P + l_{max} + N_P = 521 \times 218 + 521 + 218 = 114317$$

and the minimum TAM width required to achieve this test time lower bound is W_{min}

$$\begin{aligned} W_{min} &= \lceil \frac{L + \max(N_I, N_O)}{l_{max}} \rceil \\ &= \lceil \frac{7 \times 500 + 30 \times 520 + 9 \times 521 + \max(417 + 72, 324 + 72)}{521} \rceil \\ &= \lceil \frac{23789 + 489}{521} \rceil = \lceil \frac{24278}{521} \rceil \end{aligned}$$

$$= 47$$

Therefore, any increase after 47 bits in the TAM width does not affect the test time.

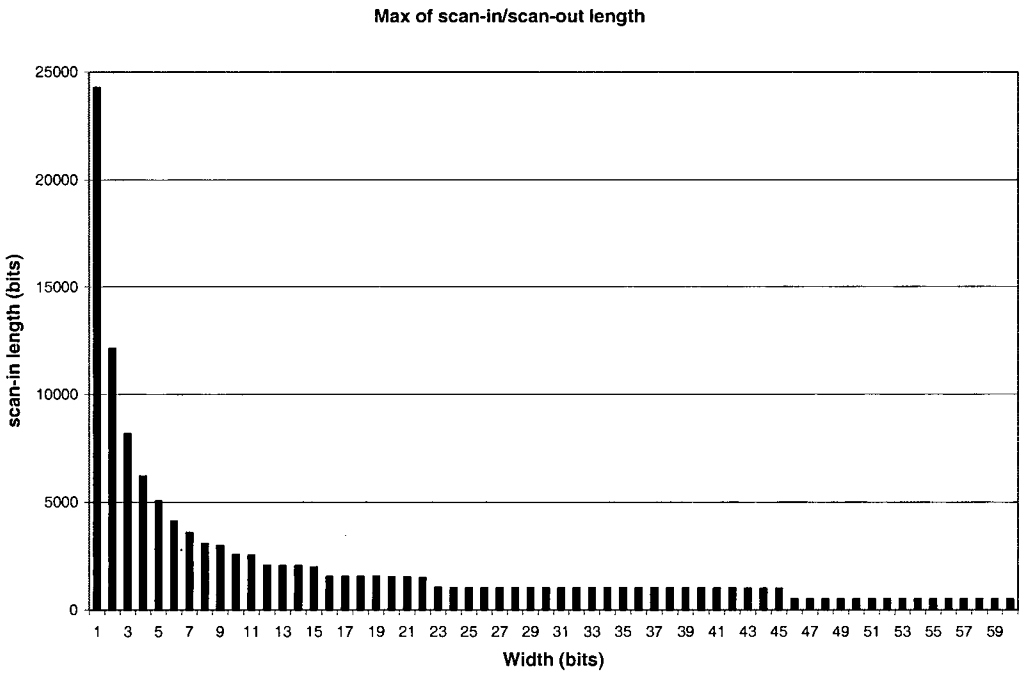


Figure 3.3: The scan-in length for core 6 in P93791 SoC from SoC benchmarks ITC'02.

3.3 Wrapper Design w.r.t. Area

As we mentioned in the previous section, sometimes a decrease in TAM width does not affect the core test time. For example, for core 6 of SoC p93791, the test time for widths 38 and 24 are the same, so we can save 14 bits on

TAM width. Therefore, our wrapper design strategy is to first minimize test time, and second, to identify the maximum number, W', of TAM chains that actually need to be created to minimize test time, when W TAM lines are provided to the wrapper. The set of values of W' corresponding to the values of $1 \leq W < \infty$ is known as the set of *pareto-optimal* points [1] for the graph.

$\mathcal{P}_\mathcal{W}$, the two-priority wrapper optimization problem, addressed in this section, can now be formally stated as follows:

- $\mathcal{P}_\mathcal{W}$: Given a core with N_I functional input, N_O functional output and N_S scan chains of lengths $l_1, l_2, \ldots, l_{N_S}$, and TAM width W, assign the $N_I + N_S + N_O$ TAM chain elements to $W' \leq W$ TAM chains, such that (i) core test time in Equation 3.1 is minimized and (ii) W' is minimum subject to priority (i).

The second priority is based on the earlier observation that test time can be minimized even when the number of designed TAM chains is less than W. This reduces the width of a TAM required to connect to the wrapper. Therefore, $\mathcal{P}_\mathcal{W}$ is analogous to the problem of MPS (minimizing the makespan of scheduling) with the secondary priority of bin-packing (minimizing the number of bins). When W' is fixed at the value of W, this problem is equal to the partitioning of TAM chain elements (PTE), which has been proved to be NP-hard (Theorem 3.2.3). Therefore, the $\mathcal{P}_\mathcal{W}$ problem is also NP-hard. To solve $\mathcal{P}_\mathcal{W}$ an algorithm with three steps is developed:

1. Calculate the possible minimum test time ($T(W)$) using TAM width W. For this step we can use the previous algorithm of wrapper design.

2. Calculate the possible minimum W', for which $W' \leq W$ and $T(W) = T(W')$. For this step we can use a standard algorithm for bin-packing, such as First Fit Decreasing (FFD). Therefore, $W' = FFD(T(W))$.

3. Now, we know the minimum test time and the minimum obtainable width. The last step is wrapper design (assignment of TAM scan chain elements, i.e., internal scan chains, inputs and outputs) for a TAM width of W' using wrapper design algorithms is discussed in the Section 3.2.3.

For example, for the core 6 of SoC p93791, when $W = 38$, our wrapper design algorithm works as follows. First, the possible minimum test time for the TAM width 38 should be calculated. Using the wrapper design algorithm (Algorithm 2), the scan-in and scan-out times are $s_i = s_o = 1040$ and the total test time ($N_P = 218$) is $T(38) = 227978$.

In the second step, we calculate the possible minimum W'. For this step we can use the first fit decreasing procedure (mentioned in Section 3.2.3). The result becomes $W' = FFD(1040) = 24$.

The last step is wrapper design for $W' = 24$, as shown in Figure 3.4. For this step we can use Algorithm 2 (for internal scan chain assignment) and then Algorithm 4 (for adding wrapper input/output cells).

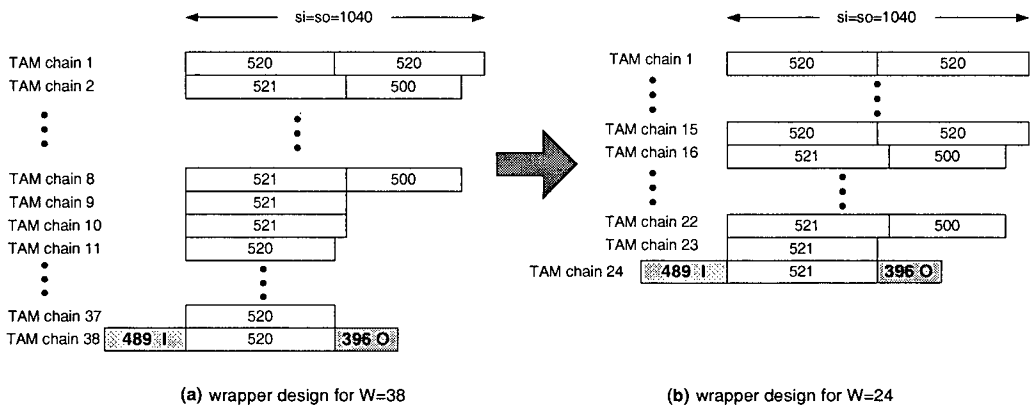


Figure 3.4: Wrapper design for core 6 in P93791 SoC from ITC'02 for TAM of width 38.

As we can see in Figure 3.4, even with a 14-bit decrease in width, the scan-in/scan-out lengths and so the core test time has not been changed.

With this algorithm we can save in the TAM width of each core, which decreases the total TAM lines required to test the whole SoC. This translates to silicon area and test I/O pins savings.

So far, the wrapper design optimizing both test time and area (silicon area and I/O pins) has been addressed. The remaining important factor in wrapper design, ATE memory, is addressed in the next section.

3.4 Wrapper Design w.r.t. the ATE Memory

In this section we discuss wrapper design optimization considering ATE resources. One of the important ATE resources which has a large impact on

ATE cost is memory. Therefore, the goal is to minimize the memory requirement for testing by eliminating useless test data memory for SoC testing. First, useless test data memory is analyzed. Next, the core wrapper design algorithm is combined with a new test vector deployment procedure stored in the ATE. To reduce memory requirements, the proposed core wrapper design finds the minimum number of TAM chain element partitions, minimizing the useless memory allocation in each partition, which facilitates efficient usage of ATE capabilities. Furthermore, the new test vector deployment procedure provides seamless integration with the ATE [24].

Since modern SoCs may consist of highly complex cores, the memory requirements to test these cores tend to increase. Therefore, careful management of memory requirements is of importance. Although tester accuracy has improved by 12% annually, IC performance increases 30% per year [12]. This forces semiconductor manufactures to step down from functional testing paths and adopt design-for-test (DFT) solutions for SoC. These new solutions, however, come with different requirements for the ATEs. Hence, the features of these new DFT-ATEs differ significantly from the conventional ATEs [10]. For example, one of the features of these new testers manages memory as a reconfigurable pool, that is, by using the right memory management, the DFT makes a larger number of transfers on some pins, while others remain unchanged [10]. Here, we show how core wrapper design and a simple ATE deployment procedure (which also accounts for memory management) can be combined to exploit this feature of DFT-ATE, leading to reduced ATE memory requirements.

Two factors influence ATE memory requirements: TAM design and core wrapper design. The previously described approaches to designing core wrappers indirectly influence ATE memory requirements. A potential straightforward solution to the ATE memory problem is a built-in self-test (BIST). However, to make existing cores BIST-ready, considerable redesign effort is required and performance penalties are incurred. An alternative solution to the ATE memory problem is test data compression [17, 37]. While test data compression reduces useful data, our goal here is to reduce useless data (in multiple scan chain embedded cores). The useless data can be explained as follows. The test vectors should be augmented with useless data (to account for the unequal lengths of multiple scan chains) to reduce on-chip control. For example, for three scan chains of lengths 2, 5 and 3, the test tools pad the scan patterns with "don't cares" (X) to make them all of equal lengths. This is illustrated in Figure 3.5, where the padded values (shown as X) are part of the test vector. Hence, they have to be stored explicitly in the ATE memory, even though they do not represent any valuable test information. This extra memory is defined as useless memory allocation, hereafter referred to as UMA [24].

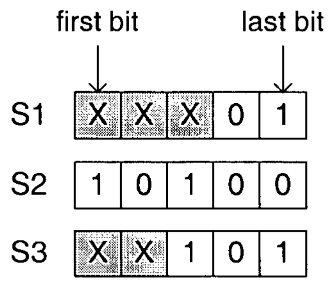


Figure 3.5: Useless Memory Allocation.

Here, the core wrapper design and the ATE memory management problems have been merged. The core wrapper design algorithm should be capable of finding the minimum number of TAM chain elements partitions such that for each partition the UMA is minimized. The ATE deployment procedure (proposed in [24]) provides memory management for the ATE, since it exploits the features of the core wrapper and deploys test data on each TAM chain partition as required. That is, the ATE deployment procedure uses the reconfigurable memory pool (RMP) feature of DFT-ATEs.

To recognize the feasibility of this method, one should note that the ATE supports sequencing-per-pin, which replicates a sequence of events in the same manner as an IC timing/logic simulator [51]. The only requirement of the ATE is to allow the memory management module to transfer test data on some ATE channels, while the remaining ATE channels remain unchanged. Recently [48], the sequencing-per-pin ATEs have been used in a similar manner where separate groups of ATE channels run at different frequencies, with each group having different memory requirements.

From Equation 3.1 we know that core test time is a function of $\max(s_i, s_0)$ because the last term of the equation, $\min(s_i, s_0)$, has a small influence on the overall test time. It should also be noted that each TAM line is assumed to be assigned to one ATE channel. Having explained the s_i and s_o, the following two examples clarify their relation to UMA.

Example Figure 3.6 shows two core wrappers with 4 inputs, 4 outputs, 4 internal scan chains of length 5, 8, 11 and 12, and a TAM with width of 4. The two designs are equivalent from the test time point of view since the longest scan-in/out have an equal length (for both designs $s_i = s_o = 12$). The TAM chains representation and the corresponding ATE memory bitmap (AMB) for both designs are shown. Because the inputs are loaded last, they are shown at the end of the memory bitmap. Considering that the ATE can control the load of each group of TAM chains of different lengths, the UMA in both cases can be reduced. However, the control overhead implied in this situation should be considered. The main difference between AMB for design 1 and design 2 is that the latter has a smaller number of ATE channels partitions (e.g. AMB1 has 4 partitions, the maximum possible number of partition in which each TAM chain is one partition, while AMB2 has just two partitions: p1 =$\{$TAM chains 1 & 2$\}$, p2 = $\{$TAM chains 3 & 4$\}$). The advantage of this (having less partition) is explained as follows.

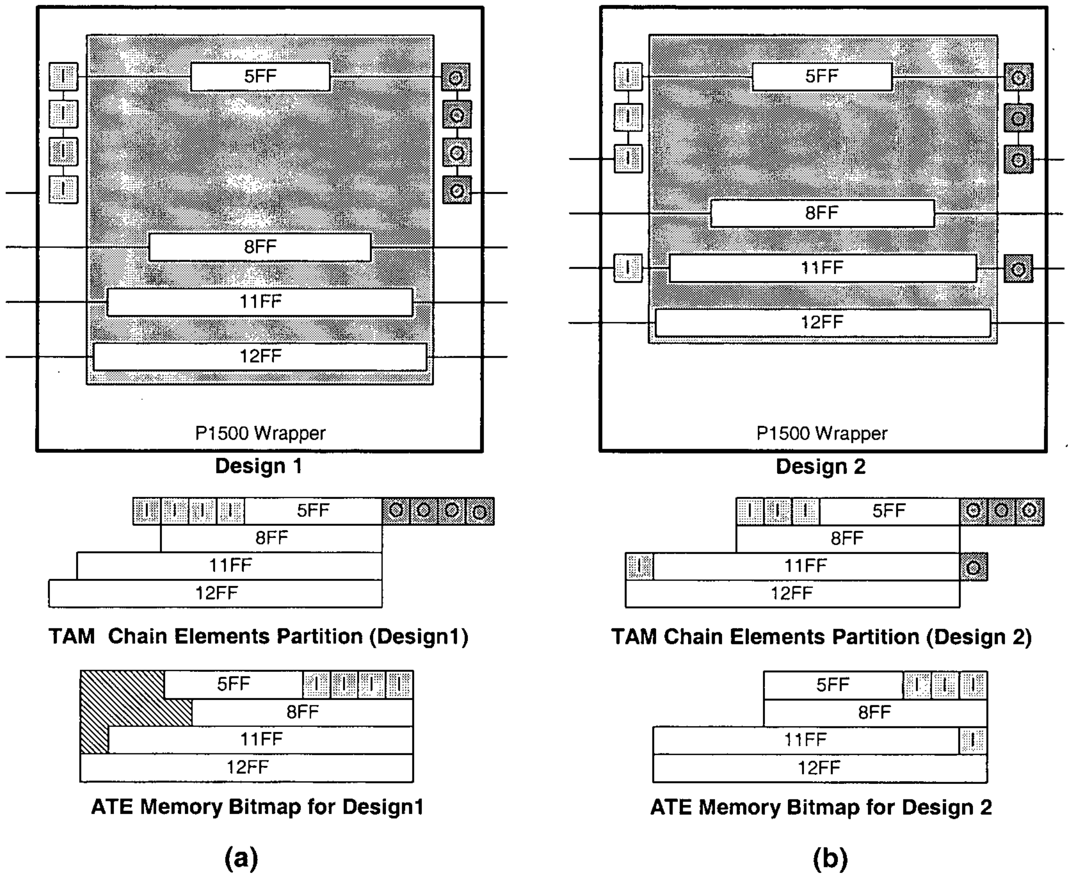

5FF
8FF
11FF
12FF
P1500 Wrapper
Design 1
Design 2
TAM Chain Elements Partition (Design1)
TAM Chain Elements Partition (Design 2)
ATE Memory Bitmap for Design1
ATE Memory Bitmap for Design 2
(a)
(b)

Figure 3.6: Alternative wrapper designs with equal test time.

Assuming a DFT-ATE with RMP, the control overhead implied by the first solution is larger because a larger number of partitions have to be controlled. Furthermore, as the number of TAM chains differ from core to core and the number of partitions obtained can vary for different cores, different number of parameters are required for the memory management of each core. On the contrary, AMB2 is shaped such that it can be easily split into two partitions (TS1 and TS2). This reduces the control, and having only two partitions, the memory management becomes very simple (Section 3.4.2). The same reasoning is applicable for sequencing-per-pin ATEs. It should be noted that the amount of test control for the first solution is clearly larger than for the second solution. This is why the first solution is considered to have useless memory as illustrated in Figure 3.6. Since the proposed ATE deployment procedure can handle maximum two partitions, for the case with more than two partitions, there will be UMA (this is one resource for UMA).

All the TAM chains are loaded in parallel using the same clock. Also, in the explanation of Equation 3.1, we note that the scan-out operation overlaps the scan-in. Hence, in a case where the number of inputs is larger than the number of outputs, the ATE memory has to account only for the UMA caused by the input scan chain, as in the previous example.

The second source of UMA is caused by the difference in TAM chain lengths when $s_i < s_o$. Since the scan-out operation of one pattern overlaps with the scan-in of the subsequent test pattern, in a case where $s_i < s_o$, we need to scan-in more data, causing useless data in ATE memory. We will show that the UMA caused by a difference of s_i and s_o cannot be eliminated completely.

Example Consider the core in the Figure 3.7(a) with $N_I = 4$, $N_O = 6$ and four scan chains of lengths 12, 11, 8 and 5. An optimum TAM chain design with respect to test time leads to $s_i = s_o = 12$. The ATE memory bitmap (AMB) of this design is shown in Figure 3.7(b) and (c).

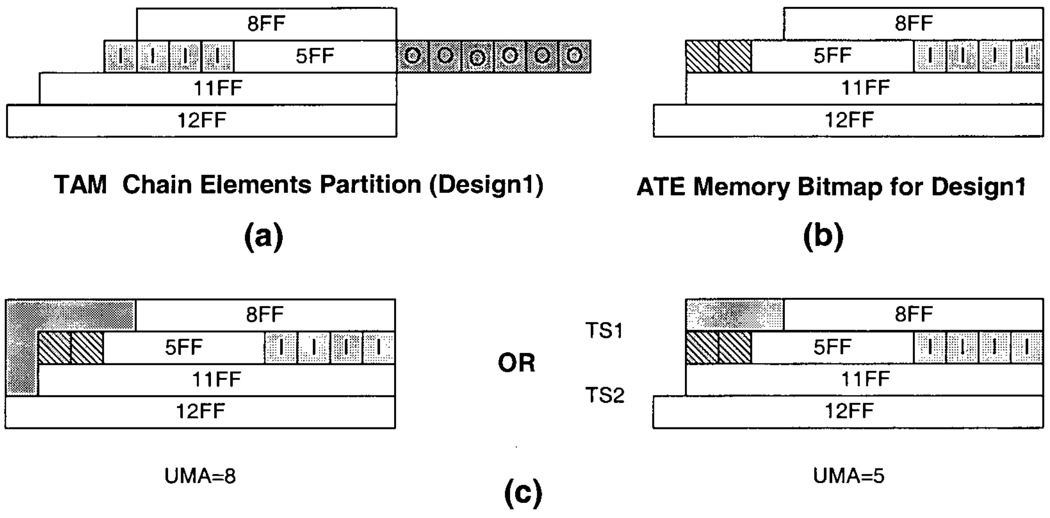
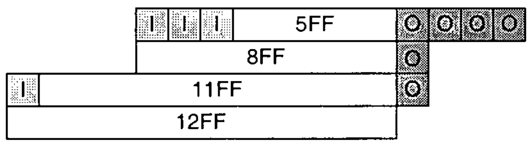


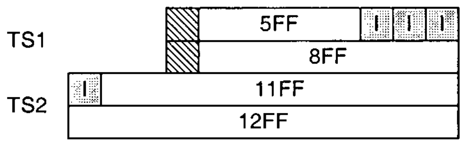
Figure 3.7: TAM chain design and ATE memory bitmap when $s_i < s_o$.

While the UMA for this design (even if we use the ATE feature) is 5, we can get a UMA of only 2 by simply rearranging the inputs and outputs in TAM chains. The design 2, shown in Figure 3.8(a), has the minimum possible UMA that we can reach for this core ,i.e., 2. The AMB for this



TAM Chain Elements Partitions (Design 2)

(a)



ATE Memory Bitmap for Design 2

(b)

Figure 3.8: TAM chain design and ATE memory bitmap when $s_i = s_o$.

design is shown in Figure 3.8(b).

Theorem 3.4.1 *For a core with N_I inputs, N_O outputs and N_S scan chains of lengths $l_1, l_2, \cdots, l_{N_S}$, and $N_I < N_O$, the lower bound on UMA is given by $N_O - N_I$.*

Proof In the case of $N_I < N_O$, s_i is smaller than s_o. Certainly then, there is a source for the UMA in the difference between s_i and s_o, even if we could prevent other UMA sources. Therefore,

$$\begin{aligned} UMA &\geq \sum_{i=1}^{i=W} (s_o - s_i) \geq \sum_{i=1}^{i=W} s_o - \sum_{i=1}^{i=W} s_i = (L + N_O) - (L + N_I) \\ UMA &\geq N_O - N_I \end{aligned}$$

where l_i is the length of scan chain i and $L = \sum_{i=1}^{N_S} l_i$. ∎

There are two steps to follow in order to reduce UMA. First, a new core wrapper design algorithm is needed. Second, the ATE has to exploit the features of new wrappers. Since the test set is divided into two, the ATE has to deploy the test vectors for the two sets at different intervals. Consider the example of Figure 3.6. The intervals at which the ATE deploys the test vectors are shown in Figure 3.9. Three parameters for ATE test vector deployment are introduced: MAX_{TC}, $Diff$ and *sp*. MAX_{TC} is the length of the maximum partition, $Diff$ is the difference between the lengths of the two partitions, and *sp* (split point) is the number of TAM chains that AMB fits to the first partition (their data should be sent as the first test set). So, for example in Figure 3.9, $MAX_{TC} = 12$, $Diff = 4$ and $sp = 2$. Since the core wrapper design is an intermediate step in a SoC test, the proposed approach does not incur any extra overhead. Hence, the ATE modifications are the only changes. For both DFT-ATEs with RMP and sequencing-per-pin ATEs, the deployment of two sets can be achieved at the expense of an external module [13] to support custom ATE behavior, employed when IEEE P1500 compliant SoCs are tested. If DFT-ATEs with RMP are employed, a test vector deployment procedure (see Section 3.4.2) can be part of the memory management. If sequencing-per-pin ATEs are considered, the methodology proposed in [48] can be used, and the control of test vector deployment can be carried out by a procedure similar to the one in Section 3.4.2.

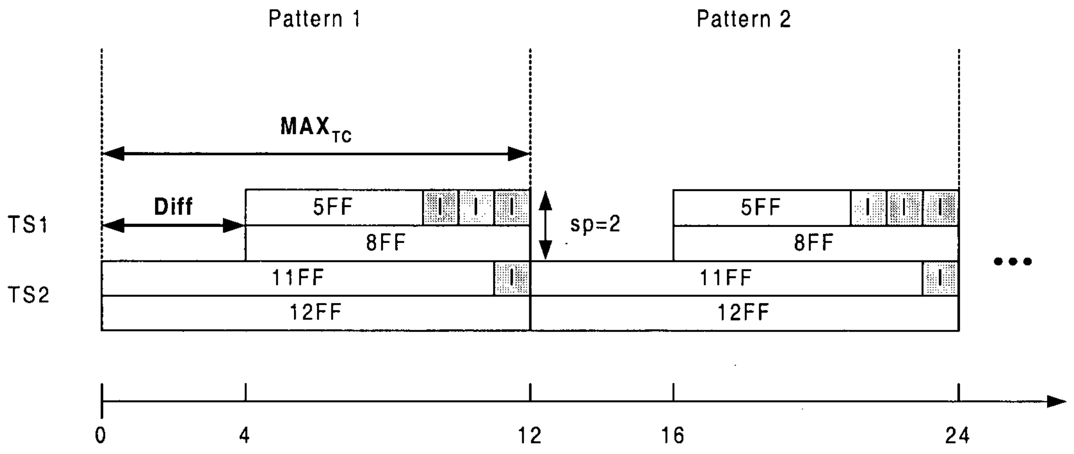


Figure 3.9: ATE test vector deployment.

3.4.1 Wrapper Design Algorithm to Reduce UMA

Here, a new core wrapper design algorithm which accounts for test time, TAM width, and UMA is proposed. Previous heuristics [32, 34, 35, 42]always aim at minimizing test time only taking into account the number of inputs or outputs. In contrast, the proposed algorithm, based on observations from the second example of Section 3.4, takes into account both the number of inputs and the number of outputs to reduce UMA. The core wrapper design problem can be formulated as follows.

Wrapper Design Problem: *For a core with N_I inputs, N_O outputs, N_S scan chains with lengths $l_1, l_2, \ldots, l_{N_s}$ and a TAM width of W, find the wrapper design for TAM width W' such that (i) core test time [in Equation 3.1] is minimized, (ii) W' is the minimum subject to priority (i), and (iii) UMA is minimized.*

Note that in this problem, test time is prioritized, that is, first we try to minimize the test time. Then we try to minimize TAM width. This problem is a $\mathcal{P}_{\mathcal{W}}$ problem defined in Section 3.4 with an additional constraint for the assignment of wrapper input/output cells. We consider UMA minimization only for Steps 2 and 3 of wrapper design, because with TAM width minimization, there are no options for the assignment of core internal scan chains to TAM chains. Here, when assigning inputs and outputs, we aim at minimizing UMA. It should be noted that the problem of wrapper design with these three priorities is NP-hard. This can be easily shown by assuming that the number of partitions equals the TAM width. In this particular case, there is no UMA and the problem reduces to the $\mathcal{P}_{\mathcal{W}}$ problem presented earlier which proved to be NP-hard. However, as illustrated in two examples in Section 3.4 (and shown again in Section 3.4.2), the number of partitions influences the complexity of the ATE program. Hence, finding the minimum number of partitions is important. Toward this goal, Algorithm 5 (Table 3.7) was developed for the assignment of wrapper input/output cells to TAM chains, giving a minimum UMA.

Note that the assumption of $N_I \geq N_O$ in Algorithm 5 (Table 3.7) is made because in UMA minimization, it is important to consider adding inputs when $N_I \geq N_O$, and adding outputs when $N_I < N_O$. Thus, when $N_I < N_O$, the outputs are added to TAM chains, then inputs. Other cases remain the same, except we add outputs first and then inputs. Reconsider the first

Table 3.7: Adding wrapper input/output cells to minimize UMA.

```
Algorithm 5: UMA Minimization
        Adding wrapper input/output cells to TAM chains
Assume $N_I \geq N_O$
sort $\mathcal{P}$ such that $l(P_1) \geq l(P_2) \geq \ldots \geq l(P_{W'})$

Adding Inputs:
Add input to the $P_1$ such that scan-in length of $P_1 = si$;
$C_1 = s_i$; (first cluster)
Initially set i:=1 and j:=2;
While $j \leq W'$
    j:=i+1;
    Try to put $P_j$ in cluster i;
        If Impossible Then $(C_{i+1}) := l(P_j)$;
        Else Add inputs to the $P_j$ such that scan-in length of $P_j = C_i$;
            j:=j+1;

Adding Outputs:
until all the outputs are assigned:
For z:=1 to $W'$
    Add outputs to the $P_z$ such that # of outputs of $P_z$ = # of inputs of $P_z$ ;

Calculating UMA, and ATE deployment parameter:
If $N_{CL} > 2$ then:
    Choose m, $1 \leq m \leq N_{CL}$, such that
        UMA:=$(C_1 * m + C_{m+1} * (N_{CL} - m)) - \{L + max(N_I, N_O)\}$ be minimum
    $UMA = (C_1 * m + C_{m+1} * (N_{CL} - m)) - \{L + max(N_I, N_O)\}$;
    $MAX_{TC} = C_1$;
    $Diff = C_1 - C_{m+1}$;
    $sp = N_{CL} - m$;
else
    If $N_{CL} = 2$:
        $UMA = 0$;
        $MAX_{TC} = C_1$;
        $Diff = C_1 - C_2$;
        $sp = |C_2|$;
    If $N_{CL} = 1$:
        $UMA = 0$;
         There is no need to use the ATE feature;

return UMA and ($MAX_{TC}$, Diff, sp) for ATE deployment procedure
```

example of Section 3.4 (p. 49). The minimum width is the same, so $W' = 4$. There are some aspects of partitioning of scan chains; first, because the number of partitions (W') is the result of a bin-packing problem, it certainly cannot be more compressed. Second, because the partitions are the result of a scheduling problem, the partitions are of the same size. Therefore, making the minimum number of clusters in this case is not hard.

Algorithm 5 first gets the W' partitions from the results of assigning core internal scan chains to W' TAM chains, and then sorts them in descending order. For the example of page 49, $p1 = \{12\}$, $p2 = \{11\}$, $p3 = \{8\}$ and $p3 = \{5\}$. Hence, we make the first cluster of size s_i which here is 12, so $C_1 = 12$. Next, we try to make each partition fit in this cluster. With adding one input, p_2 can fit in cluster1, because the scan-in length of this TAM chain will be 12 again. When we try the same thing for p_3, we note that it is impossible to fit p_3 in C_1, because we need 4 inputs, though only 3 inputs are left. Therefore, we make another cluster of size 8, $C_2 = 8$. Next, we start from the following partition, p_4, to make it fit. By adding 3 inputs to p_4, it fits in the second cluster. Therefore, the result of Algorithm 5 for this example is two clusters of size 12 and 8. The number of clusters (N_{CL}) varies from one (best case: even the new feature of ATE is not required because all the TAM chains have the same length) to W' (means even two partitions could not fit into one cluster). It is highly desirable that the number of clusters be one or two. To have a single cluster, $max(N_I, N_O) = max(s_i, s_o) * W' - \sum_{i=1}^{N_S} l_i$, which is not usually the case. This makes a double cluster the most desirable and practical situation. In [24], it is shown that even though for $N_{CL} = 2$ the UMA is not always 0, this particular case leads to a good solution from an UMA standpoint, with the advantage of eliminating extra ATE requirements. Therefore, at the end of Algorithm 5, we check whether the number of clusters is more than two, and we try to fit all the clusters into two new clusters. If the number of clusters is more than two, $N_{CL} > 2$, the UMA is no longer zero, but because the ATE procedure is simple, and the UMA is less than in a case where the ATE features are not used, it is worthwhile. To try to fit N_{CL} clusters into just two clusters, we make the first new cluster the largest, $C'_1 = C_1$. The second new cluster could be any of the clusters which minimizes the UMA: $C'_2 = C_{m+1}$, $2 \leq (m+1) \leq N_{CL}$. In this case $|C'_1| = m$ and $|C'_2| = N_{CL} - m$. The UMA is computed as the difference between the area of two new clusters and the sum of the area of all the clusters, i.e., UMA $= (C'_1 * |C'_1| + C'_2 * |C'_2|) - (\sum_{i=1}^{N_{CL}} |C_i| * C_i)$. For the above example $N_{CL} = 2$, so the UMA is zero and ATE deployment procedure parameters are $MAX_{TC} = 12$, $Diff = 12 - 8 = 4$, and $sp = 2$.

3.4.2 ATE Deployment Procedure to Reduce UMA

In order to fully exploit the new core wrapper design, the initial test set is divided into a number of sets equal to the number of partitions. The ATE program must deploy test vectors from different sets at separate times. Hence, an increase in the number of partitions leads to a more complex ATE program. However, if the number of partitions is limited to two, the necessary changes in the ATE deployment are minor. This section gives the pseudo-code for the ATE program for this particular case (two partitions).

Consider two clusters C'_1 and C'_2 that are the result of Algorithm 5. Since $|C'_1| + |C'2| = W'$ (this is how the partitions are constructed) and $C'_1 > C'_2$ (the TAM chains are in descending order, see Algorithm 5). As mentioned in the explanation of Algorithm 5 (p. 58), $MAX_{TC} = C'_1$, $Diff = C'_1 - C'_2$, and $sp = |C'_2|$. Using this information, the initial test set can be divided into two sets. The deployment of the test vectors at different intervals can be easily achieved by supplying the ATE with the three parameters MAX_{TC}, $Diff$, and sp in addition to the two test sets (TS1 and TS2). The pseudo-code for a very simple ATE procedure, which accounts for the parameters mentioned above, is shown in Table 3.8. This procedure takes as inputs the width of the test bus W', the number of patterns N_P, and the three parameters MAX_{TC}, $Diff$, and sp. During MAX_{TC} clock cycles, TS2 is loaded onto the test bus from ATE memory. Since the first partition is smaller than the second, the

Table 3.8: ATE deployment procedure.

Procedure 1

```
$i = N_P$
While $(i > 0)$ do
    for $j := 1$ to $MAX_{TC}$
        if $i > Diff$ then
            load[$1 \cdots sp$]=read-mem(TS2,$N_P \cdot Diff + j$)
        load[$sp \cdots W'$]=read-mem(TS1,$N_P \cdot MAX_{TC} + j$)
    $i = i - 1$;
```

ATE reads the test data for the TS1 in only $Diff$ clock cycles.

3.5 Wrapper Design for Flexible Cores

For cores with flexible-length scan chains, (1) input wrapper cells, (2) core-internal scan flip flops, and (3) output wrapper cells can be equally distributed over the available TAM wires. This is easy because the length of all elements is one. Hence, for a core with N_I inputs, N_O outputs, and N_F internal scan flip flops (for non-scanned cores, $N_F = 0$) when the TAM with width W is assigned to it, the scan-in length $s_i(W)$ and scan-out length $s_o(W)$ can be defined as follows [42]:

$$s_i(W) = \lceil \frac{N_F + N_I}{W} \rceil \tag{3.2}$$

$$s_o(W) = \lceil \frac{N_F + N_O}{W} \rceil \tag{3.3}$$

Thus, the test time for the core can be calculated from Equation 3.1.

By dividing TAM chain elements equally, we actually minimize the scan-in and scan-out lengths, so we minimize test time.

Actually the only difference in all the theorems and equations for fixed-length cores is that for the flexible cores, the lengths of all the scan chains are one, so $l_{max} = 1$ and $L = \sum_{i=1}^{N_S} l_i = N_F$. If we apply Theorems 3.2.1 and 3.2.2 for flexible cores, the following result is derived.

Corollary 3.5.1 *For a flexible core with N_I inputs, N_O outputs, N_F scan flip flops, and N_P test patterns, the upper bound on the core test time, T, is given by*

$$T_{upper} = (1 + N_F + \max(N_I, N_O))N_P + N_F + \min(N_I, N_O)$$

Corollary 3.5.2 *For a flexible core with N_I inputs, N_O outputs, N_F scan flip flops, and N_P test patterns, the lower bound on the core test time, T, is given by $T_{lower} = 2N_P + 1$ and the minimum TAM width required to achieve this lower bound on the test time is given by $W_{min} = N_F + \max(N_I, N_O)$.*

The second step of wrapper design, as with fixed cores, is TAM width reduction. Here, we look for a smaller width than W that gives the same test time. Note that since the first term of Equation 3.1 is more important, by "same test time", we mean "same $\max(s_i, s_o)$". Therefore, we look for the possible minimum $W' \leq W$, such that $\max(s_i(W), s_o(W)) = \max(s_i(W'), s_o(W'))$.

Theorem 3.5.3 *Given a core with N_I functional inputs, N_O functional outputs and N_F scan flip flops and a TAM with width W, there is $W' \leq W$ with $\max(s_i(W), s_o(W)) = \max(s_i(W'), s_o(W'))$; the upper bound on W' is*

$$W' = \lceil W - \frac{W - ((N_F + max(N_I, N_O))\%W)}{max(s_i(W), s_o(w))} \rceil^{‡}$$

Proof Assume that $W' = W - m$, $m \geq 0$, and m is the maximum possible integer number such that $\max(s_i(W), s_o(W)) = \max(s_i(W'), s_o(W'))$. From Equations 3.2 and 3.3

$$\max(s_i(W), s_o(W)) = \lceil \frac{N_F + max(N_I, N_O)}{W} \rceil$$

Letting $A = N_F + max(N_I, N_O)$, $B = \max(s_i(W), s_o(W))$ and $C = (N_F + max(N_I, N_O))\%W = C$,then $B = \lceil \frac{A}{W} \rceil = \lfloor \frac{A}{W} \rfloor + 1$, so $A = (B-1)W + C$. Also, we know that $\max(s_i(W-m), s_o(W-m))$ should be the same for W. So

$$\max(s_i(W), s_o(W)) \quad = \quad \max(s_i(W-m), s_o(W-m))$$

‡ *$m\%n$ "m modulo n."*

$$\begin{aligned}
B &= \lfloor\frac{A}{W-m}\rfloor+1\\
B &= \lfloor\frac{(B-1)W+C}{W-m}\rfloor+1\\
B &= \lfloor\frac{(B-1)W+C+(B-1)m-(B-1)m}{W-m}\rfloor+1\\
B &= \lfloor\frac{(B-1)(W-m)+C+(B-1)m}{W-m}\rfloor+1\\
B &= \lfloor B-1+\frac{C+(B-1)m}{W-m}\rfloor+1\\
B &= B-1+\lfloor\frac{C+(B-1)m}{W-m}\rfloor+1\\
B &= B+\lfloor\frac{C+(B-1)m}{W-m}\rfloor\\
0 &= \lfloor\frac{C+(B-1)m}{W-m}\rfloor
\end{aligned}$$

Hence, $C+(B-1)m < W-m \Rightarrow m < \frac{W-C}{B}$.

That is $m = \lfloor\frac{W-((N_F+max(N_I,N_O))\%W)}{\max(s_i(W),s_o(W))}\rfloor$. We know that $W' = W-m$, therefore $W' = \lceil W - \frac{W-(N_F+max(N_I,N_O))\%W}{max(s_i(W),s_o(w))}\rceil$ ∎

The next step in wrapper design is to minimize the UMA. Actually, our strategy of dividing scan flip-flops and wrapper input/output cells equally to TAM lines not only minimizes the test time, but also minimizes the UMA. We will show that for flexible cores the UMA is either zero or the minimum possible number.

If we divide all the TAM chain elements equally, we have two clusters: $C_1 = \lceil\frac{N_F+max(N_I,N_O)}{W'}\rceil$ and $C_2 = \lfloor\frac{N_F+max(N_I,N_O)}{W'}\rfloor$. So, the following parameters of ATE memory bitmap are derived:

- $MAX_{TC} = \lceil \frac{N_F + max(N_I, N_O)}{W'} \rceil$.
-

$$Diff = \begin{cases} 0, & ((N_F + max(N_I, N_O)) moduloW) = 0 \\ 1, & Othrewise \end{cases}$$

- $sp = W' - ((N_F + max(N_I, N_O)) moduloW)$
-

$$UMA = \begin{cases} 0, & N_I \geq N_O \\ N_O - N_I, & Otherwise \end{cases}$$

- Required Memory=$N_P \cdot (MAX_{TC} \cdot W' - sp \cdot Diff)$.

To summarize, for wrapper design of a flexible core, first we find the smallest W' and then divide all the TAM chain elements, including scan flip-flops and wrapper input/output cells equally on the W' TAM chains.

The experiments have been done on circuits from the ISCAS'89 benchmarks.

Example Consider the circuit *S1196* from ISCAS'89 benchmarks [2] with $N_I = N_O = 14$, $N_F = 18$ and $N_P = 113$ [26].

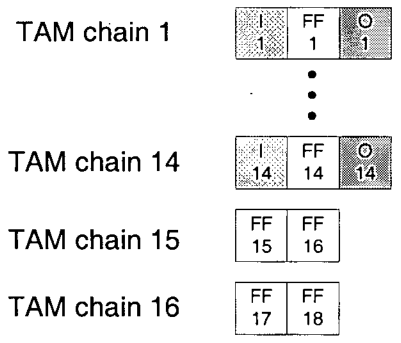
Assume that we assign a TAM with width 28, $W = 28$. In the first step, we should find W'.

From the core information, we can calculate $N_F + max(N_I, N_O) = 32$ and so $max(s_i(W), s_o(W)) = \lceil \frac{N_F + max(N_I,N_O)}{W} \rceil = \lceil \frac{32}{28} \rceil = 2$.

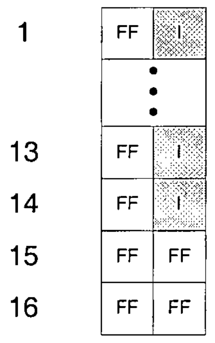
Now, we can calculate W' from Theorem 3.5.3.

$$\begin{aligned}
W' &= \lceil W - (\frac{W - (N_F + max(N_I, N_O))\%W}{\max(s_i(W), s_o(W))} \rceil \\
W' &= \lceil 28 - \frac{28 - (32\%28)}{2} \rceil \\
W' &= \lceil 28 - \frac{28 - 4}{2} \rceil = \lceil 16 \rceil \\
W' &= 16
\end{aligned}$$

The next step is to divide the TAM chain elements equally into W' TAM lines; the result is shown in Figure 3.10(a).



(a) TAM chain Elements Partitions



(b) ATE Memory Bitmap

Figure 3.10: Wrapper design for S1196 from ISCAS'89 benchmarks.

Also, the AMB for the wrapper is shown in Figure 3.10(b). The ATE memory bitmap parameters are as follows: $MAX_{TC} = 2$, $Diff = 0$, $sp = 16$,

$UMA = 0$ and required memory is 3616 bits .

The wrapper design algorithm has been applied for different widths. The results are reported in Table 3.9. For all the widths, the UMA is zero, because here $N_I \geq N_O$.

Table 3.9: The result of the wrapper design algorithm for S1196 of ISCAS'89 benchmarks.

W	W'	s_i	s_o	Test Time (cycles)
1	1	32	32	3761
2	2	16	16	1937
3	3	11	11	1367
4	4	8	8	1025
5	5	7	7	911
6	6	6	6	797
7	7	5	5	683
8-10	8	4	4	569
11-15	11	3	3	455
16-31	16	2	2	341
≥ 32	32	1	1	227

3.6 Experiments

In previous sections (3.2-3.4.1), the important issues of wrapper design have been discussed. To validate the efficiency of the proposed wrapper design method, several experiments were performed on the large cores of the ITC'02 SoC test benchmarks [43].

Consider Core 6, the largest logic core from p93791. In Section 3.3, we presented a wrapper design for this core for $W = 38$, and showed that the width can be decreased to $W' = 24$ (Figure 3.4(b)). The last step of our wrapper design is UMA minimization. The final result of this wrapper design is shown in Figure 3.11(a).

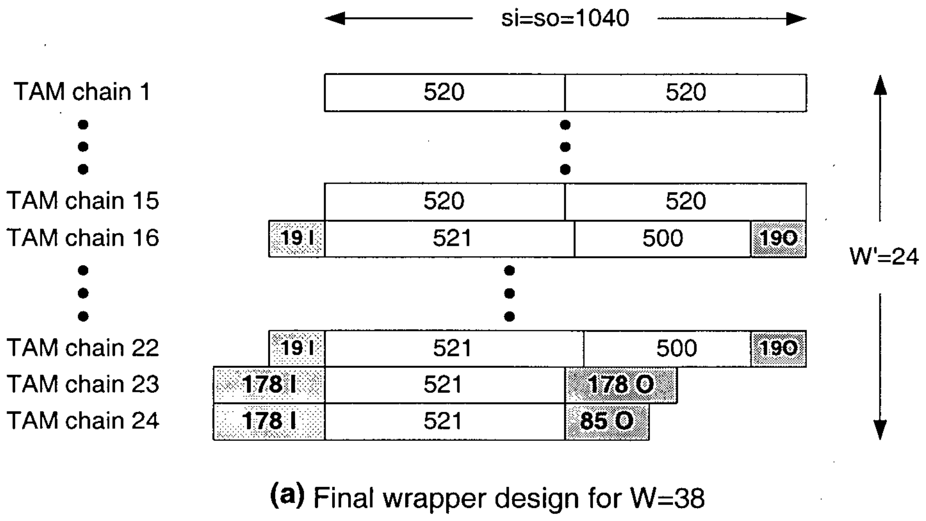
The ATE Memory bitmap of the final design has been shown in Figure 3.11(b). The UMA for this design is zero, and we saved 14 bits in the TAM width (W'=24).

The wrapper design results for all widths for core 6 of p93791 is reported in Table 3.10. There are many noteworthy points. First our algorithm minimizes UMA for most cases (UMA is often zero) while it saves TAM width and test time. Other researchers only attempt to minimize test time [42], or to minimize test time and TAM width [32–35], or to minimize test time and UMA [24]. As far as we are aware, no previous work considers all these aspects to date.

In Table 3.10, we also see that the wrapper design algorithm performs better than all other algorithms. It saves up to 77% in the ATE memory, 58% in TAM width and 2% in test time.

The TAM width and ATE memory for my method and previous method are compared in Figures 3.13 and 3.12, respectively.

Core 6 of SoC P93791, was the largest core in all the cores of the ITC'02 SoC Test benchmarks. A lot of experiments on different SoC of the ITC'02 SoC Test benchmarks and also ISCAS'85 and 89 benchmarks were run. For example the test time vs. width for core 3 to core 10 of the SoC D695 is shown in Figure 3.14.



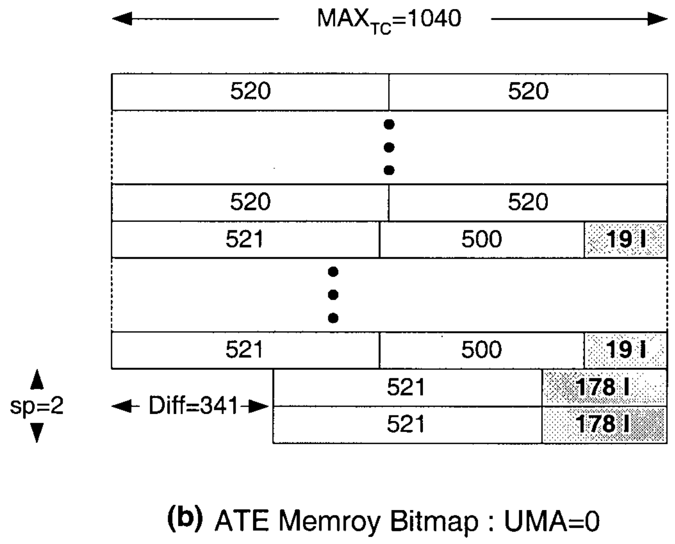


Figure 3.11: The final wrapper design for core 6 of p93791 from ITC'02 SoC test benchmarks for TAM width=38.

Table 3.10: The result of the wrapper design algorithm and comparisons with previous works for core 6 of p93791 of the ITC'02 SoC Test Benchmarks

W	W'	s_i	s_o	Test Time	UMA	Required Memory	Mem. impr. (%)[a]	W impr. (%)[b]	Time impr. (%)[c]
1	1	24278	24185	5317007	0	5292604	0	0	0
2	2	12139	12093	2658613	0	5292604	0	0	0
3	3	8180	8180	1791638	0	5292604	1.1	0	1.01
4	4	6202	6202	1358456	0	5292604	2.2	0	0
5	5	5060	5060	1108358	2	5293040	4.2	0	1.62
6	6	4140	4140	906878	0	5292604	2.3	0	0.02
7	7	3620	3620	792998	0	5292604	4.4	0	0.03
8	8	3100	3100	679118	0	5292604	2.2	0	0.03
9	9	3000	3000	657218	6	5293912	11.2	0	2.7
10	10	2580	2580	565238	2	5293040	6.3	0	0.04
11	11	2560	2560	560858	3	5293258	16	0	0.04
12	12	2080	2080	455738	0	5292604	2.8	0	0
13-14	13	2060	2060	451358	0	5292604	10.3	7.7	0.05
15	15	2000	2000	438218	10	5294784	23.5	0	2.05
16-19	16	1560	1560	341858	0	5292604	2.8	18.8	0
20-21	20	1540	1540	337478	46	5303068	26.6	5	0
22	22	1500	1500	328718	2	5293040	35.9	0	1.4
23	23	1056	1052	231478	0	5292604	0.04	0	0
24-38	24	1040	1040	227978	0	5292604	2.8	58.3	0
39-42	39	1020	1020	223598	14	5295656	63.8	7.7	0
43-45	43	1000	1000	219218	2	5293040	77.1	4.7	0
46	46	528	526	115848	0	5292604	0.04	0	0
≥ 47	47	521	521	114317	0	5292604	0.9	-	0

[a]Memory savings comparing to [32]
[b]TAM width savings comparing to [24, 42](maximum improvement)
[c]Test Time savings comparing to [32]

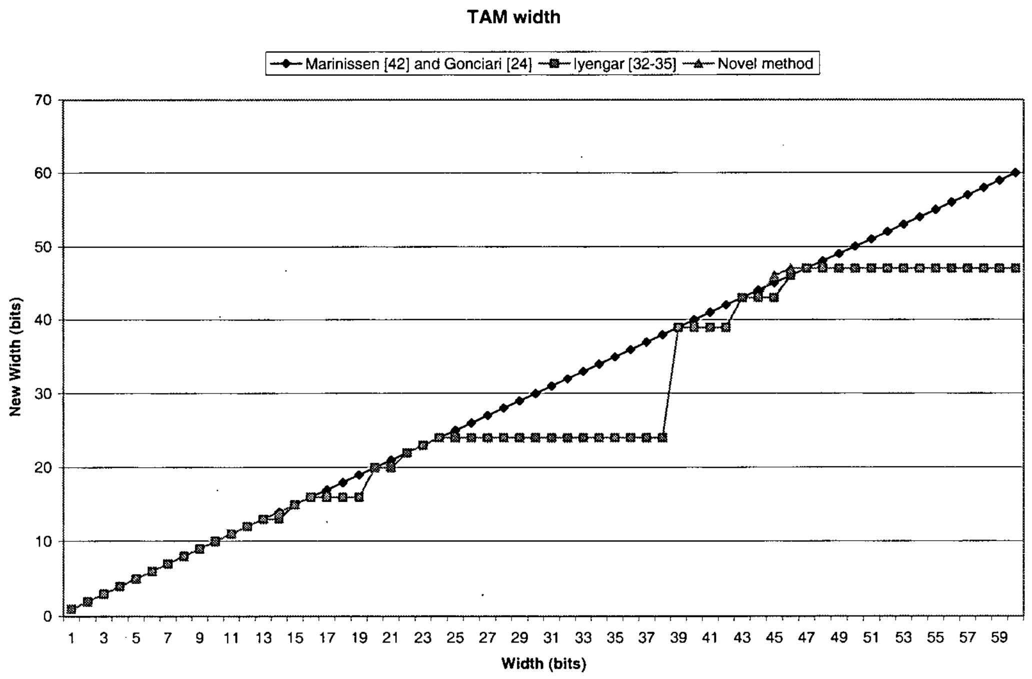


Figure 3.12: New TAM width for core test vs. given TAM width for core 6 in P93791 SoC from SoC benchmarks ITC'02.

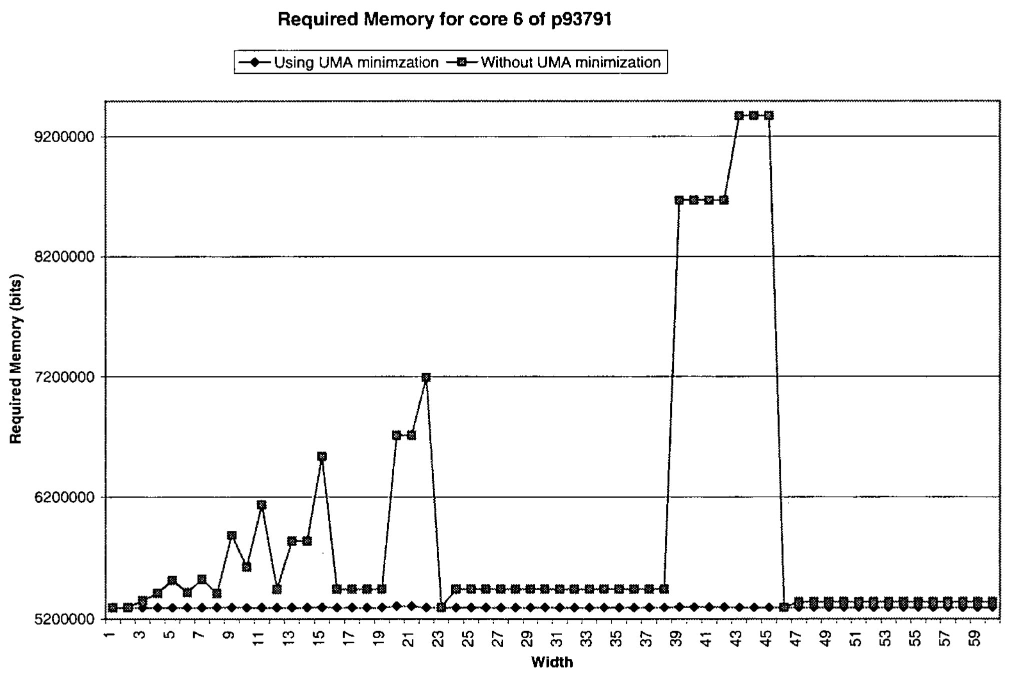


Figure 3.13: Required memory for core test vs. TAM width for core 6 in P93791 SoC from SoC benchmarks ITC'02, with and without UMA minimization.

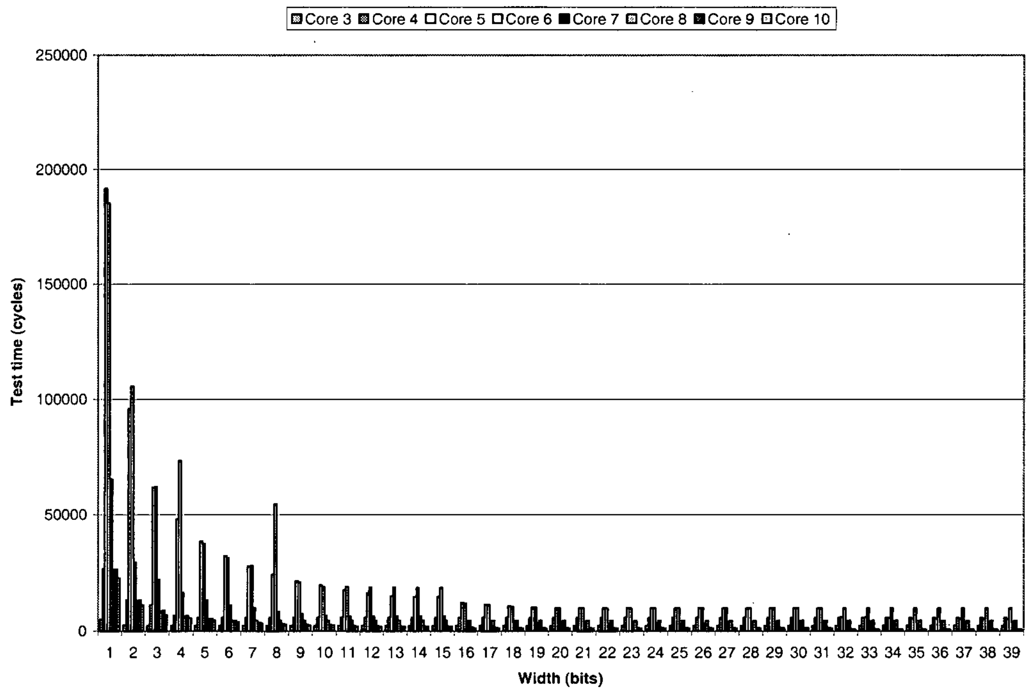


Figure 3.14: Test time vs. TAM width for core 3 to 10 in D695 SoC from SoC benchmarks ITC'02.

3.7 Conclusions

In this chapter, we present a novel wrapper design algorithm which can be used as a tool for SoC designers (integrators). The proposed algorithm considers all test costs. The algorithm can handle cores with both fixed-length scan chains and flexible-length scan chains. The experiments show improvement in comparison to existing methods.

Chapter 4

Time Domain Multiplexed TAM

In this chapter a novel TAM based on time domain multiplexing (TDM) is introduced. TDM-TAM not only has all the advantages of common bus-based TAMs, such as scalability and efficiency in time and area, but it is also flexible, reconfigurable and handles cores with BIST efficiently. It also needs less global routing than common bus-based TAMs. The underlying concepts of TDM-TAM are presented and test time and area models are derived. A dynamic masking method is described. It gives the tester the flexibility to change test scheduling and strategy after fabrication. Also, an example of how to use TDM-TAM in testing multi-frequency SoCs and a TAM optimization method for TDM-TAM are reported. Finally, a comparison is drawn between TDM-TAM and other types of TAM.

4.1 Basics

TDM-TAM is a bus-based TAM that uses logic situated locally at each core to enable or disable the core, such that bus contention cannot occur. This architecture eliminates the necessity for global address lines, which are normally required in a bus-based architecture. In our TDM-TAM, the data to be sent on the TAM bus is divided into frames. Each core is assigned a specific mask enabling the cores to extract the appropriate data bits from the frames.

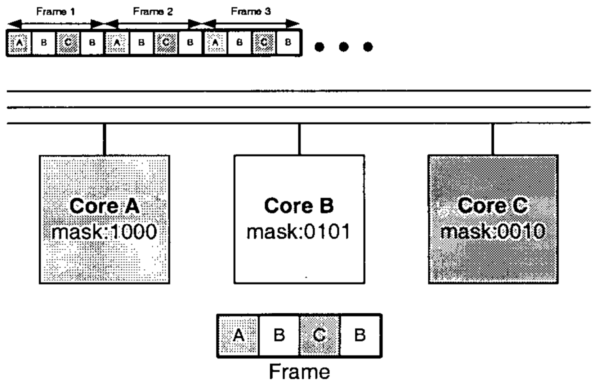


Figure 4.1: Example illustrating the concept of Time Domain Multiplexing (TDM).

For example, in Figure 4.1, a frame is assumed to consist of four bits. Core **A** uses the first bit of each frame, core **B** uses bits 2 and 4, and core **C** uses bit 3. For this specific configuration, assume that we wish to send the data "11" to core **A**, "00" to core **B** and "don't care" bits "XX" to core **C**. The two required frames to be sent on the TAM bus in this case must, therefore, be "10X0", followed by "1XXX". Optimal frame and mask assignment is obviously critical for the efficiency of the scheme. This is addressed later in this chapter.

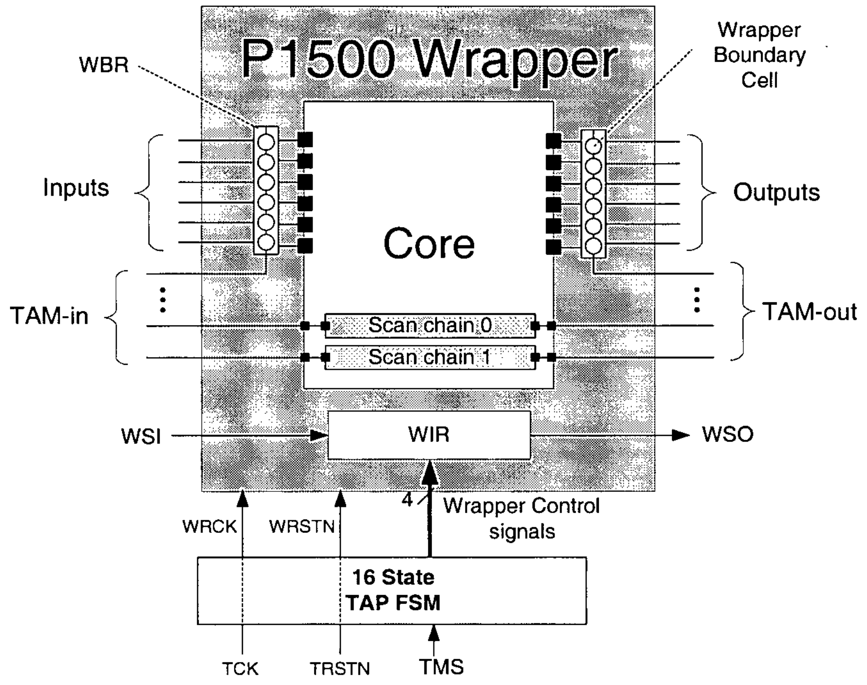


Figure 4.2: Conceptual view of the IEEE P1500 wrapper and TAP controller.

To implement the TDM-TAM, a P1500 wrapper is used to wrap each core. Figure 4.2 shows the P1500 wrapper and the 1149.1 TAP controller.

In the standard 1149.1 TAP controller, a 16-bit finite state machine generates the wrapper control signals from the serial TMS * bit stream. For our TDM-TAM, we transform the original 1149.1 TAP controller into a TDM-

*Test Mode Select

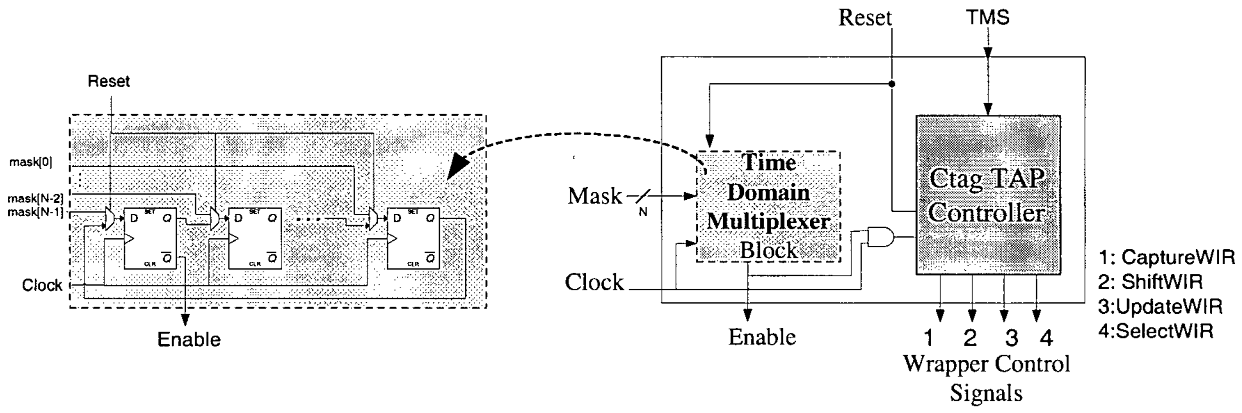


Figure 4.3: Block diagram for TDM-TAM TAP Controller.

TAP controller (Figure 4.3). This controller includes the original 1149.1 FSM as well as some minimal extra logic, referred to as a Time Domain Multiplexer (TDM) Block, which creates the *Enable* signal for the core according to a preassigned mask.

Here, we assume that the masks are assigned, before the layout and Fabrication step. In this case, the mask for each core is said to be local. This means the mask lines for each core are hard-wired to ground or power supply. However, in Section 4.4, we show that it is useful to have the ability to change a mask after fabrication. We refer to this feature as dynamic masking.

The TDM block operates as follows. When the *Reset* signal goes low, the mask is loaded into the flip flop chain. When the *Reset* is high, the mask is shifted by one bit on every falling edge of the clock. Using the falling edge allows the *Enable* signal to be stable before the rising edge of the clock, thus avoiding glitches when used to gate the clock. The *Enable* signal generated for each core is used to enable three different components. When a core's *Enable* signal is active (high), the TMS control signal is read by the core's TAP controller and the core's P1500 wrapper. The core is then clocked, allowing it to read data from the *WSI* and *TAM-in.* Moreover, the tri-state buffer is enabled, thereby allowing the core to write to the *WSO* and *TAM-out.* When *Enable* is deactivated (low), all these components are deactivated, allowing other cores on the same TAM bus to use the bus. Cores tested by a full BIST scheme may not be disabled, so that their test can be executed as quickly as possible, although their TAP controller, P1500, and tri-state buffer may still be deactivated. Test power considerations may, however, require that BISTed cores be disabled at times as well.

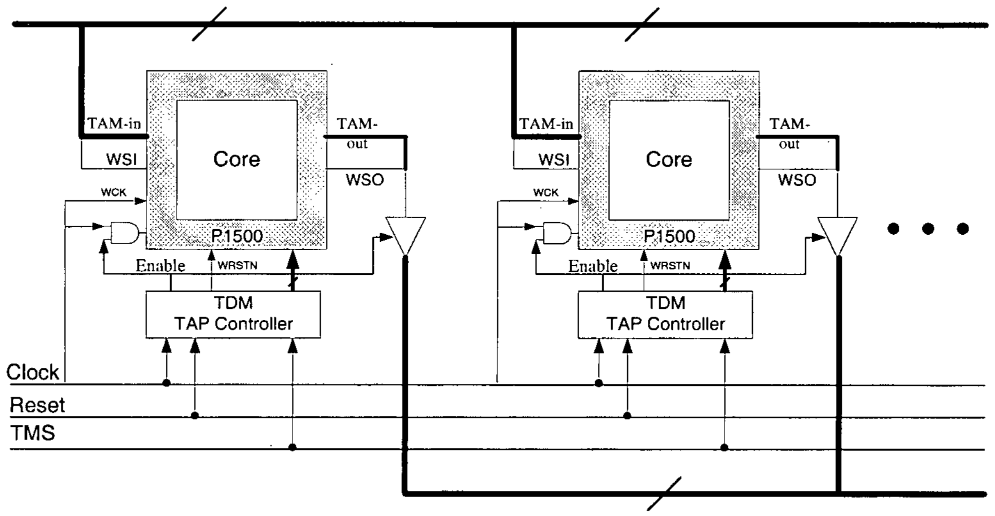


Figure 4.4: Single branch TDM-TAM.

Figure 4.4 illustrates what we refer to as a *single branch* TDM-TAM with multiple cores, where a branch includes the one-bit wide TMS, a one or more bit-wide *TAM-in* bus (to accommodate at least the *WSI* signal) and a one or more bit-wide *TAM-out* bus. Hence, the minimum data line width of a branch is three. Each branch also includes global *Clock* and *Reset* signals. All the cores connected to a branch have a local TDM-TAP controller and associated logic, and a tri-state buffer. They share *TMS, Clock, Reset, TAM-in,* and *TAM-out* lines.

Different SoCs can have different numbers of TDM-TAM branches. A case where all the cores connect to the same branch constitutes the simplest, *single-branch* case, illustrated above. The other extreme is to have as many branches as there are cores on the SoC, with each core connected to its own private branch.

4.2 Timing Model

In this section, a model for test time with the TDM-TAM is developed. To refine the test time model, we consider the time required to send instructions to the wrapper, and to put the wrapper in test mode, as well as the time required for loading/capturing a signature (if necessary). We assume that the cores are tested using either test patterns provided externally via the TAM, or by using patterns generated and evaluated within the core, that is, by using a form of built-in self test (BIST). For BISTed cores, the TAM is only assumed to send a form of "start BIST" instruction, and to subsequently communicate a BIST result, such as a signature. The cores that are not BISTed require not only test instructions, but also test pattern data to be sent via TAM. Based on the latter observation, we define two parameters for each core: t_I and t_D. t_I is the portion of the core test time during which the core does not need to control or use the TAM, and is therefore independent of the mask assignment. For instance, the time a BISTed core requires for the BIST test patterns to be generated, applied and evaluated. t_D is the part of a core's test time that requires the control/use of the TAM and which is, therefore, dependent on the mask assignment, that is, the time required to send a specific test instruction to a core.

Let the SoC consist of N_C cores and N_B branches, and assume that core j, $1 \leq j \leq N_C$, is assigned to branch k, $1 \leq k \leq N_B$. The t_I portion of the test time of core j, t_{I_j}, does not depend on the mask assignment. Hence the core-branch connections do not effect this part of the core test time. On the other hand, the t_D portion of the core test time depends on the cores' mask assignments and frame lengths, that is, t_{D_j} depends on mask assignments and frame lengths. The example in Figure 4.1, makes clear that as the number of "ones" in a core's mask increases (i.e., with increasing Hamming weight), the proportion of data from each frame used by the corresponding core increases accordingly. Hence, the test time for core j, when assigned to branch k, is:

$$T_{jk} = \lceil \frac{t_{D_j}}{||mask_j||} \rceil * M_k + t_{I_j} \tag{4.1}$$

Where M_k is the number of bits in a frame for branch k and $||mask_j||$ is the number of ones in the mask assigned to core j. To illustrate, reconsider the example in Figure 4.1 and assume that core **B** is not BISTed and that the TAM-independent test time is 20 cycles while $t_{D_B} = 15$. From Figure 4.1, M for the bus is 4, $||mask_A|| = ||mask_C|| = 1$ and $||mask_B|| = 2$. Core **B** can use two bits of every 4-bit frame. Therefore, testing core **B** requires $\lceil\frac{15}{2}\rceil$ time frames. In turn, this implies a total of 8×4 clock cycles.

Let X_{ij} be a 0-1 variable defined as follows:

$$X_{jk} = \begin{cases} 1, & \textit{core } j \textit{ is assigned to bus } k \\ 0, & \textit{Otherwise} \end{cases}$$

Using Equation 4.1, the test time for each core can be determined such that total test time for testing all cores assigned to bus k amounts to $T_k = max_{j=1}^{N_C}(X_{jk} * T_{jk})$ since all the cores can be tested concurrently due to *time division multiplexing.* Assuming all TAM branches can be used simultaneously, the total test time amounts to $T = max_{j=1}^{N_B}(T_k)$ or

$$TT = \max_{k=1}^{N_B}(\max_{j=1}^{N_C}(X_{jk} * T_{jk}) \tag{4.2}$$

Theorem 4.2.1 *For a SoC using TDM-TAM and with N_C cores, an upper bound on the total test time is given by*

$$UP_{TT} = \max_{i=1}^{N_C} U_i \cdot M$$

where U_i is the upper bound of the test time for core i given by Theorem 3.2.2.

Proof The single branch with the minimum width TDM-TAM is the worst case for test time of an SoC. To derive an upper bound on the test time, assume that all cores of SoC lie on one branch having a one-bit wire width for TAM-in/TAM-out. This implies that the test time of each core equals the corresponding upper bound for test time. Assuming that the worst case for test time is $\max_{i=1}^{N_C} U_i$, and that such a core is not BISTed, then t_I of this core is zero and $t_D = \max_{i=1}^{N_C} U_i$. Furthermore, in the worst case, such a core has a minimum share of the bus. That is, it is assigned only one bit in each frame. Under these conditions, the total SoC test time reaches its upper bound: $TT = \lceil \frac{t_D}{1} \rceil * M + t_I = \max_{i=1}^{N_C} U_i * M$. ∎

Theorem 4.2.2 *For a SoC using TDM-TAM with N_C cores, a lower bound on the total test time is given by*

$$LO_{TT} = \max_{i=1}^{N_C} L_i$$

Where L_i is the lower bound on the core i test time given by Theorem 3.2.1). TDM-TAM width required to attain the minimum test time is

$$W_{LO} = \sum_{i=1}^{i=N_C} W(T_i \leq LO_{TT})$$

Proof The minimum test time is attained when each core has its own branch ($N_B = N_C$) wide enough to test each core at its lower bound test time, so the test time of each core should be the minimum possible test time, L_i (given by Theorem 3.2.1). Hence, the total test time, based on Equation 3.1, is $\max_{i=1}^{N_C} L_i$.

Also, in this configuration each branch width can be calculated from the minimum width required to achieve a core test time equal to or less than the minimum total test time of SoC. Therefore, the TDM-TAM width required for a minimum total test time is $W = \sum_{i=1}^{i=N_C} W(T_i \leq LO_{TT})$. ∎

4.3 Overhead Area Model

Modeling area for the TDM-TAM is straight forward since the same additional logic is required for each core. In our area modeling, we neglect the wiring area. The overhead for each core is due to the TDM-TAP controller and a tristate buffer. Hence, for an SoC with N_C cores, the total area overhead $A_{TDM-TAM} = N_C * (A_{buffer} + A_{TDM-TAP\,controller})$ where A_{buffer}

Table 4.1: Area estimates for TDM-TAM controller block.

Circuit	**Area**(μm^2)	**Description**
Buffer	37	tri-state buffer
CTAG-TAP	1142	CTAG TAP controller
FF & MUX	123	Flip Flop & 2×1 Multiplexer

and $A_{TDM-TAPcontroller}$ correspond to the area of the buffer and TDM-TAP controller, respectively. The TDM-TAP controller, illustrated in Figure 4.3, includes a **CTAG TAP Controller** block and a TDM block. Therefore, the overhead area is this:

$$A_{TDM-TAM} = N_C(A_{buffer}+A_{TDM-TAP}) = N_C(A_{buffer}+A_{CTAG-TAP}+A_{TDM\,block})$$

In the TDM block, there are M flip flops and M 2×1 multiplexers. Therefore, $A_{TDM\,block} = M * (A_{FF} + A_{2\times1MUX})$, where A_{FF} and $A_{2\times1MUX}$ is the area of the flip flop and multiplexer respectively. From the above, the area overhead for the scheme can be expressed as:

$$A_{TDM-TAM} = N_C(A_{buffer} + A_{CTAG-TAP} + A_{TDM\,block}) \tag{4.3}$$

Estimates of actual area for the constituent blocks appear in Table 4.1 (for a $0.18\mu m$ CMOS technology).

In Equation 4.3, the area overhead is proportional to the number of cores in the SoC and to the length of the data frames, M (also corresponding to the number of bits required to encode the mask associated with each core). As the frame length increases, the area overhead increases. However, test time generally decreases with increased frame length, thereby resulting in the usual tradeoff between area and test time.

4.4 Dynamic Masking

The preceding discussion includes an assumption that the mask associated with each core is hard-wired prior to fabrication by making appropriate ground and power line connections. However, it is possible to implement the scheme such that the masks are programmed at arbitrary post-fabrication times. We refer to this scenario as *Dynamic Masking.* Dynamic masking can be realized at the expense of little additional logic over that required for the *static* cases. Dynamic masking offers many potential advantages, primarily that of flexibility, allowing for better and more effective post fabrication test resource optimization.

One example of an advantage offered by dynamic masking is that of increased core diagnostics, deemed necessary only at post fabrication, and possibly subsequently to an initial test phase. For example, a given core under test may need to be more easily diagnosed. This may be achieved *a posteriori* by modifying the core's mask to allow for more test data to reach and leave the core per given test time. Also, dynamic masking can accommodate test data/pattern alterations that occur in post fabrication, and therefore, allow for better test resource optimization (e.g. test time minimization) subsequent to such changes and fabrication. Dynamic masking can also be exploited to modify the mask assignments to accommodate different core test times or data requirements for cores connected on the same or different branches. Finally, power dissipation, coupling, and other noise or performance related post-fabrication effects can be more easily handled by virtue of the versatility introduced by dynamic masking.

Dynamic masking can be implemented by adding a new state in the TDM TAP controller that controls the shifting in of a mask to the flip flop chain of the TDM block in Figure 4.3. Note that for realizing dynamic masking, the only additional logic required in the TDM block is a multiplexer. This multiplexer needs to be controlled by the **TAP Controller** to select between the WSI (for shifting in a new mask) and the feedback signal (for normal operation).

4.4.1 Multi-Frequency SoC

Today's SoC could have cores with different frequencies. None of the existing TAMs are suitable for multi-frequency SoC testing. The proposed TDM-TAM can handle the multi-frequency SoC, because with the TDM technique, we can manipulate frequency. Consider the example in Figure 4.1: data fed to core **A** and core **C** is a quarter of the frequency of the branch. Core **B** is fed by a branch frequency. Also, with n-lines and a multiplexer, we can have a line with frequency n times the frequency of a single line.

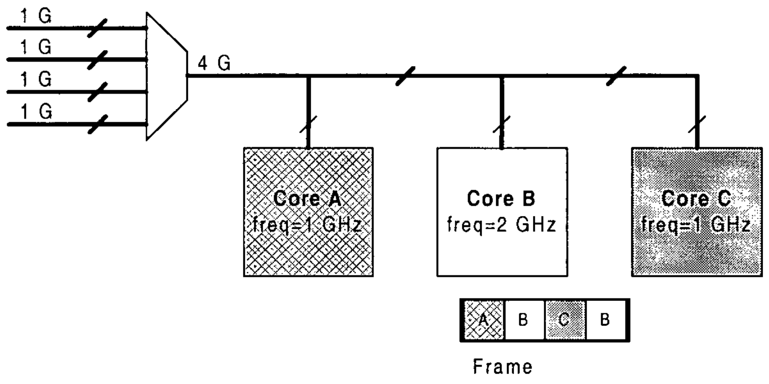


Figure 4.5: Using TDM-TAM for multi-frequency SoC.

Figure 4.5 shows a SoC with 3 cores, working at two different frequencies. Assume that ATE can send at 1 GHz. In Figure 4.5, the one-branch configuration that can test each core at its own speed is shown. The mask of core **A**, **B** and **C** should be “1000”, “0101” and “0010” respectively. This way, if the frequency of the branch is **F**, the frequency of buses going to core **A**, **B** and **C** is $\frac{\mathbf{F}}{4}$, $\frac{\mathbf{F}}{2}$ and $\frac{\mathbf{F}}{4}$. Based on this formulation, **F** should be 4 GHz, but the maximum frequency of the ATE machine is 1 GHz. Using 4 lines and a multiplexer, we can generate a 4 GHz line. With this configuration, we can test each core at its own speed. Please note that Figure in 4.5 does not show TDM-TAM in full detail.

4.5 Optimization

In previous sections, we describe the basic concepts of TDM-TAM, as well as a test time model, and a model for area overhead. One of the most important issues associated with TAM architectures is SoC test time minimization. This section focuses on this issue, and assumes that core designs and their test requirements are fixed. Specifically, we focus on the optimal assignment of cores to test buses and the optimal assignment of masks to individual cores, assuming the TDM-TAM architecture is in place.

We address the following problems: *Mask Assignment*: Assuming an SoC using the TDM-TAM scheme and N_C cores is assigned to N_B branches, determine the optimal mask assignment for each core.

Core-Branch Pairings: Assume an SoC using the TDM-TAM scheme and a total of N_C cores is assigned to N_B branches; determine the optimal core-branch pairing for each core.

These problems are in fact revised from earlier TAM optimization works [14, 19] where the objective is to find the optimum configuration for a specific TAM [41], to achieve a minimum test time. In our special TDM-TAM, the problem is not only (1) finding the best assignment of cores to buses, but also (2) finding the best mask assignments for each core.

Toward solving these optimization problems, here we used a Genetic Algorithm (GA)-based method. Our program requires the following information

Table 4.2: U226 from ITC SoC'02 benchmarks.

Core	No. of primary I/Os	No. of test patterns	No. of scan chains	Scan chain lengths	TAM use
1, 2, 3	2/1	1363968	0	-	n
4, 5, 6	3/17	2666	0	-	y
7	97/64	76	20	52	y
8	34/32	1048576	0	-	n
9	17/10	15	0	-	y

as input: number of cores, number of branches, the test strategy for each core, and whether any functional (non-scan) test patterns need to be applied, the number and length of the core scan chains, and the number of core input/outputs. Our program outputs an optimal branch configuration and core mask assignments.

Example 1: Consider the U226 SoC benchmark from the ITC'02 SoC test benchmarks [7]. The characteristics of this benchmark are reported in Table 4.2. We use a TDM-TAM design for this SoC and apply our optimization. We assume two branches with a minimum width, i.e. $N_B = 2$, and the number of bits/frame to be sixteen in both cases (i.e. $M_1 = M_2 = 16$). As five of the nine constituent cores are assumed to be connected to the TAM for this benchmark, the problem is optimally assigning these five cores to two branches and making optimal mask assignments for each core. Our optimization program yields the assignment of cores 4, 5 and 9 to the first branch, and cores 6 and 7 to the second, as illustrated in Figure 4.6(a). For the first branch, the optimal mask assignment is such that $||mask_4|| = ||mask_5|| = 7$ bits, and $||mask_9|| = 2$ bits, while those for the second branch are such that $||mask_6|| = 6$ bits and $||mask_7|| = 10$ bits. This configuration and assignment yields a total test time of 140160 cycles. This test time results creates a case where the branches are both of minimal width, that is, each TAM branch consists of only one available data line, and hence, the total effective *TDM-TAM width* = 2[†]. However, when the total effective *TDM-TAM width = 3,* the optimal configuration differs. The optimal configuration is illustrated in Figure 4.6(b), that is, branch one is attributed an effective width of one bit (minimal width), with cores 4 and 5 assigned to it such that $||mask_4|| = ||mask_5|| = 8$. Branch two is attributed an effective width of two bits, with cores 6, 7 and 9 assigned to it, and such that $||mask_6 = 6||$, $||mask_7|| = 10$ and $||mask_9|| = 2$. The total test time for this configuration amounts to 95984 cycles. In Figure 4.7, the total test time (cycles) versus total effective TDM-TAM width is given, assuming the two-branch configurations shown for benchmark U226.

Example 2: In this example, we consider the D695 SoC benchmark from ITC'02 SoC test benchmarks [7]. The test data for cores in the D695 is shown in Table 4.3. In the first step of TAM design, we should apply available wrapper design algorithms, and for each core, calculate the test time for

[†]Note that we define the total effective TDM-TAM width as the total number of input/output TAM lines, excluding the necessary control lines, such as TMS, Clock and Reset.

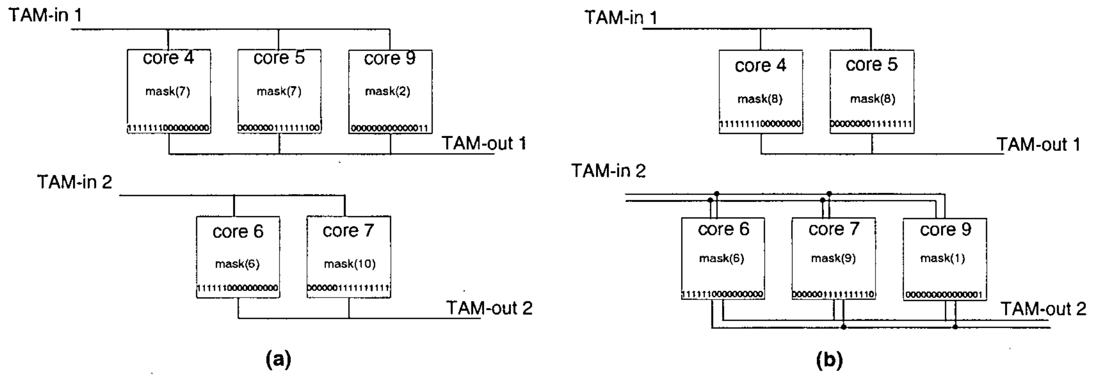


Figure 4.6: Optimal configurations for SoC U266 assuming two branches: (a) total effective TDM-TAM width=2, (b) total effective TDM-TAM width=3.

different widths. The result of using the wrapper design algorithm for the cores in D695 is reported in Table 4.4. Note that core 1 and core 2 are not scan-testable cores, so the test time for them is equal to the number of test patterns for all the widths. In Table 4.4, the test time for $w = 1$ is actually UP_T for the core, and the last number of each column is the minimum test time (LO_T), and corresponding width is W_{min} of the core. First, we try to design a one branch TDM-TAM with a minimum width for this SoC. In the worst case scenario, the core with the largest UP_T has the minimum share of the bus, so in this example, if we assume that $M = 16$, the upper bound on the SoC test time becomes $16 * max_{i=1 \dots 10}(UP_{T_i}) = 16 * 191874 = 3069984$. However, even in this case if we apply our optimization algorithm to find a better masking scheme, we can save 200% in test time. The best masking scheme (for $M = 16$) is 1, 1, 1, 1, 3, 3, 3, 1, 1, 2 for cores $1, 2, \cdots, 10$

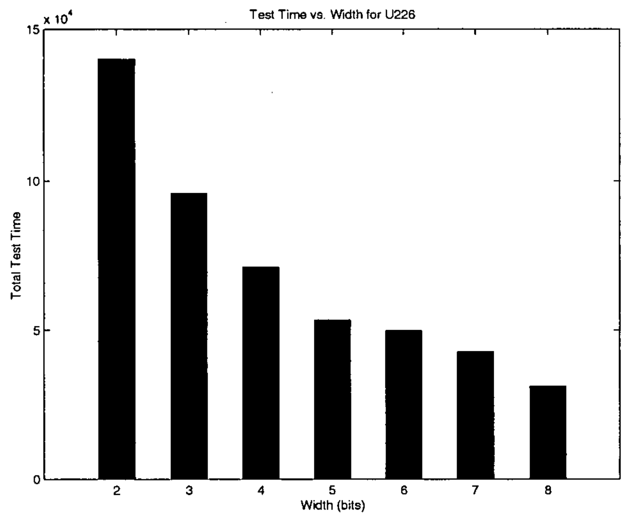


Figure 4.7: Benchmark U226: Test time (cycles) vs. total effective TDM-TAM width for two-branch configurations.

respectively, giving a test time of 1023328. As mentioned before, with the increase of M (the number of bits in the mask), the test time may decrease. For example, here, for $M = 32$, the best masking scheme is 1, 1, 1, 3, 10, 8, 3, 1, 1, 5, and the corresponding test time is 425632: 140% further improvement.

To find the lower bound on the test time for D695, as we proposed in Theorem 4.2.2, the lower bound is the maximum of all lower bounds of the cores' test time. From Table 4.4, core 6 has the largest lower bound (9869), so the lower bound on the total test time of D695 SoC is 9869. To calculate the width required to achieve this lower bound, Theorem 4.2.2 can be used: $W = \sum_{i=1}^{10} W_i(T_i < 9869) = 1+1+1+4+32+20+9+3+3+3 = 77$. For

Table 4.3: Test data for the cores in D695

Circuit (core)	No. of Primary I/Os	No. of test patterns	No. of scan chains	Scan chain length Min	Max
c6288	64	12	-	-	-
c7552	315	73	-	-	-
s838	35	75	1	32	32
s9234	75	105	4	52	54
s38584	342	110	32	44	45
s13207	214	234	16	39	41
s15850	227	95	16	33	34
s5378	84	97	4	44	46
s35932	355	12	32	54	54
s38417	134	68	32	51	55

all cores, the width required to achieve core test time ≤ 9869 is highlighted in the Table 4.2.2. In this design, each core i has its own private branch width $W_i(T_i < LO_{TT})$ and the mask of each core is M. However, we show that with a better design, the width needed to attain the LO_{TT} can be less. For example, in the following design shown in Figure 4.8, total test time is the lower bound (9869) but the total width is 65: 12 lines savings in TAM width.

Table 4.4: Test data for the cores in D695

width	**Core's Test Time**							
(W)	3	4	5	6	7	8	9	10
1	**5058**	26602	191874	185489	65381	22327	26351	22580
2	2658	13354	95992	105798	29393	11243	13182	11296
3	2507	11129	61870	62246	21926	**8691**	**8802**	**6934**
4		**6782**	48106	73400	16477	5769	6597	5660
5		5829	38478	37338	13224	4605	5310	4653
6			32164	31240	11027		4440	3990
7			27613	27964	9966		3798	3327
8			24163	54610	8341		3305	2836
9			21518	20906	7383		2964	2664
10			19646	19034	6716		2820	2664
11			17624	19034	6431		2418	2073
12			16205	18799	6431		2226	2001
13			14983	18799	6431		2118	2001
14-15			14762	18564	6431		2118	2001
16			12192	11978	4219		1659	1374
17			11420	11273	4026		1572	1338
18			10869	10571	3739		1488	1338
19			10319	10103	3549		1416	1338
20			9989	**9869**	3454		1416	1338
21-24			9989		3359		1416	1338
25-31			9878				1416	1338
32			**6206**				836	763
33			5985				822	739
34			5765				798	727
35			5655				774	
36			5435				750	
37			5325				738	
38			5215				714	
39			5105					

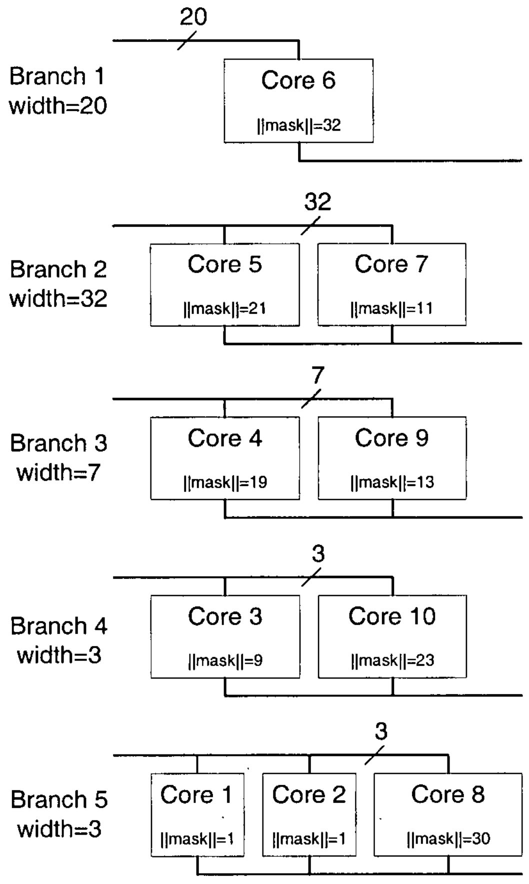


Figure 4.8: Benchmark D695: Design for minimum test time, 5 branches, with mask 32 bits and, total width=65, total test time=9869.

4.6 Case Study

4.6.1 Platform Description

To evaluate the effectiveness and tradeoffs associated with the proposed TDM-TAM, a network processor engine (NPE) design was developed and used as a target test vehicle. Our NPE is an OSI Layer 3 device that forwards IPv4 packets [29]. NPE's major blocks include a pre-processing unit, a classifier, an embedded processor, several post-processing units, and various memory components. The blocks communicate with each other through an $AMBA^{TM}$-compliant high speed bus. Point-to-point connections are also used for control signals and interrupts.

NPE blocks were developed following reuse guidelines such as I/O buffering and sub-block partitioning. The embedded processor is a modified Motorola $HC11^{TM}$ micro-controller. The AMBA-compliant bus was developed using the AMBA AHB specifications available from ARM Ltd. The bus uses a master-slave request-based architecture with multiple clock cycles per transfer. The pipelined bus design yields an efficient implementation that satisfies the high bandwidth requirements of NPE.

For the core-level test methodology, a full-scan test methodology is assumed for the preprocessing and the post-processing units. One single scan chain was created for each block, and scan vectors were generated by an ATPG tool and assumed to be provided to the blocks from an external source. To emulate a heterogeneous test methodology environment, the classifier and embedded processor blocks are assumed to be tested using a logic BIST methodology. The logic BIST uses a 32-bit linear feedback shift register (LFSR) to generate pseudo-random test vectors, and uses a signature analyzer to compact the test results. The memory modules use memory

BIST that runs a Marching C algorithm. All the blocks and the associated test structures are encapsulated with P1500-compliant wrappers [4].

As illustrated in Figure 4.9, we compared three different TAM architectures using the NPE as a target design. A first TAM that we investigated in our comparison, referred to as *Serial P1500,* leverages the new P1500-compliant wrappers. The proposed P1500 standard architecture resembles the STD 1149.1 Test Access Port and Boundary Scan architecture. The most noticeable difference is the removal of the TAP controller and the addition of a parallel test port. By removing the TAP controller and providing more access ports, the serial input constraint of STD 1149.1 is removed and a wide variety of test access mechanisms are supported. It is possible to serially thread P1500 wrappers to create a simple TAM for a SoC.

The second TAM that we included in our comparison is the Network (Novel) Indirect and Modular Architecture (NIMA) [45]. The basis of NIMA is the establishment of indirect digital communication paths between cores through the use of packet-switching connections. We used the on-chip network for test purposes. Hence, the test methodology when assuming NIMA consists of converting test vectors into packets that can find their way from a source to their destination cores. Similarly, test results can also be transferred from cores to a sink. The underlying assumption here is that all test-related communications destined for or originated from the SoC need to be communicated using the on-chip network. Finally, the third TAM used in our comparison is the TDM-TAM.

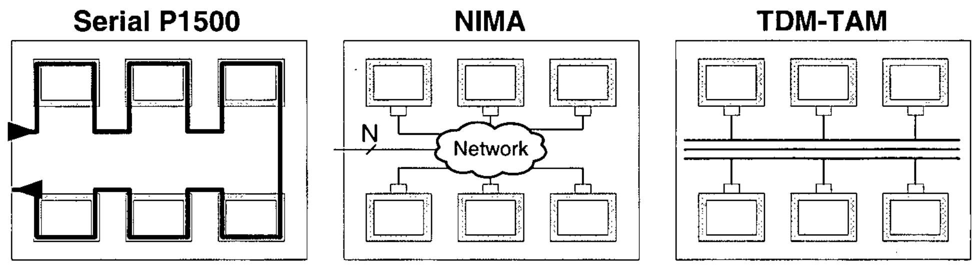


Figure 4.9: TAM architectures considered in our comparison.

While all three TAMs considered in our study use P1500 wrappers, the NIMA TAM requires that test data be pre-processed into packets. Therefore, with NIMA, test data (either scan test vectors and/or BIST instructions) must initially be converted into test packets and subsequently sent to designated blocks. In turn, to translate the test packets back into information understandable by a P1500 wrapper, a NIMA interface module is required for each SoC block. A controller within the NIMA interface module generates the P1500 wrapper control signals. Since the P1500 specifications do not include the implementation of such a controller, we assumed the use of the 1149.1 TAP controller for this purpose. This controller is responsible for generating the required P1500 control signals from the NIMA test packets such that the P1500 wrapper performs the operations outlined in the standard. Furthermore, in the case of NIMA, test data is parallelized by the test network. Hence, a *data funnel* lying between the test network and the NIMA interface modules of each core is required for down-converting the data width. Finally, since test data from the NIMA test network arrives at the NIMA core interfaces in bursts, a buffering scheme must be incorporated to ensure that the P1500 wrappers receive data only when ready to accept it.

As mentioned above, two test methodologies were applied to our NPE design, BIST and full scan. BIST has the advantage of lower test traffic compared to full scan. Moreover, BIST eliminates the need for off-chip vector storage and management, and thereby requires a simpler ATE. On the other hand, the BIST methodology used here is based on pseudo-random pattern generation, which typically suffers from reduced fault coverage, in comparison to full scan, unless special measures are taken. One way to mitigate this drawback, for example, is to boost the number of BIST test patterns in comparison to the number of full scan test patterns. This in turn amounts to longer test times. Consequently, it is evident that various test methodologies have different impacts on the choice and effectiveness of different TAM architectures.

4.6.2 Results

We derived specific test time and area overhead models for the TDM-TAM when the latter is applied to the NPE design described above. For test time, we derived expressions that depend on a core's test methodology whether full scan or BIST. For BISTed cores, we assumed that a given set of instructions are required to initiate and complete the BIST, while the actual test (application of test patterns and signature generation) can proceed independently of the specific TAM architecture. Hence, for a BISTed core, t_d is the time required to send the BIST instructions, while t_i is the core test time. Based on simulations we performed on our NPE design, the times required to implement core BIST functions are as follows:

Instruction	**Time**(Cycles)
Reset	1
Start BIST	11
Load Reference Signature	38
Capture Signature	16

Hence, for BISTed cores:

$$\begin{cases} t_d = Reset + StartBIST = 12\ cycles \\ t_i = t_c = CoreScanLength * CoreVectorCount \end{cases}$$

Cores tested using a full-scan test methodology must use the TAM throughout the entire test session. Hence, $t_i = 0$ for such cores. Based on our simulations, t_d includes a 3 times loading instruction, and once loading signature and capture signature, plus the time needed to load the test pattern here given as $VectorCount * (ScanLength + 5)$. For the cores using full-scan:

$$\begin{cases} t_i = 0 \\ t_d = 2 + 3\ LoadInstruction + LoadSignature + \\ \qquad VectorCount(ScanLength + 5) + CaptureSignature \end{cases}$$

The above formula for t_i and t_d are not general and differ for different designs. Nevertheless, our simulation with our NPE shows that these formulae are relatively accurate for test time predictions for our NPE design. Test time prediction errors appear to be bounded by M cycles in our specific case.

For the area overhead associated with TDM-TAM when applied to our NPE, we used the area model described in Section 4.3. Results for the frame lengths of 16 and 32 are reported in Table 4.5, assuming the NPE design implemented in a $.18\mu m$ CMOS technology.

The serial TAM has minimum area overhead, very close to that of TDM-TAM. That is, the test time for TDM is much shorter than that of serial TAM. The area overhead for the NIMA is very large.

Table 4.5: Overhead area comparison.

TAM Type	Overhead Area	
	μm^2	%
TDM (with 16-bit mask)	18882	0.75
TDM (with 32-bit mask)	30690	1.22
NIMA	1343236	53.3
Serial	5041	0.2

For the test time, the three TAMs were compared in six different scenarios corresponding to increasing numbers of cores tested using a full scan test methodology, that is, corresponding to increasing test data volumes (Table 4.6). Figure 4.10 illustrates the total test time (measured in clock cycles) required for testing the NPE, assuming full-scan DFT for NIMA and TDM of different widths (from 2 to 5 bits), and the serial TAM. The horizontal axis indicates the total test data (in bits) transferred. The BIST execution time is approximately 530,000 cycles for all six scenarios, and it is chosen as the threshold for total test time. From Figure 4.10, it is obvious that the serial TAM has the worst test time and TDM with width 5 has the shortest. The test time of NIMA is very close to that of TDM, but the area overhead for NIMA is much larger than that of TDM-TAM.

TDM-TAM has the disadvantages of bus-based TAM. For example, TDM-TAM performance is better for small SoCs, rather than huge SoCs (Figure 4.10). Also, the interconnect will be an important issue in future for TDM-TAM design.

Table 4.6: Scenarios for assessing TAM performance.

	Total test bits	Core that use full scan
Scenario 1	420014	pre and post
Scenario 2	522460	pre, post, and HC11
Scenario 3	692364	pre, post, HC11, and classifier
Scenario 4	972072	pre, post×4, HC11, and classifier
Scenario 5	1789122	pre, post×4, HC11×4, and classifier
Scenario 6	2769456	pre×4, post×4, HC11×4, and classifier×4

4.7 Conclusions

We proposed a new bus-based TAM which is scalable and compares favorably to other proposed TAMs in terms of test time and area requirements. The proposed TAM uses the concept of time domain multiplexing (TDM) to effectively reduce TAM area requirements, while still achieving good test time performance. This is made possible by the optimal assignment of cores to buses and the optimal assignment of time slots in the time domain multiplexing scheme. We illustrated the use of a genetic algorithm-based methodology to achieve these optimal assignments. We presented test time and area requirement models for our TDM-TAM.

TDM-TAM can be a solution for testing multi-frequency SoCs. By using TDM-TAM for testing, we can not only test each core at its own speed, but also test the SoCs without high-frequency and expensive ATEs.

TDM-TAM not only offers generally excellent time and area performance, but also can be implemented to enable optimal reconfiguration of test re-

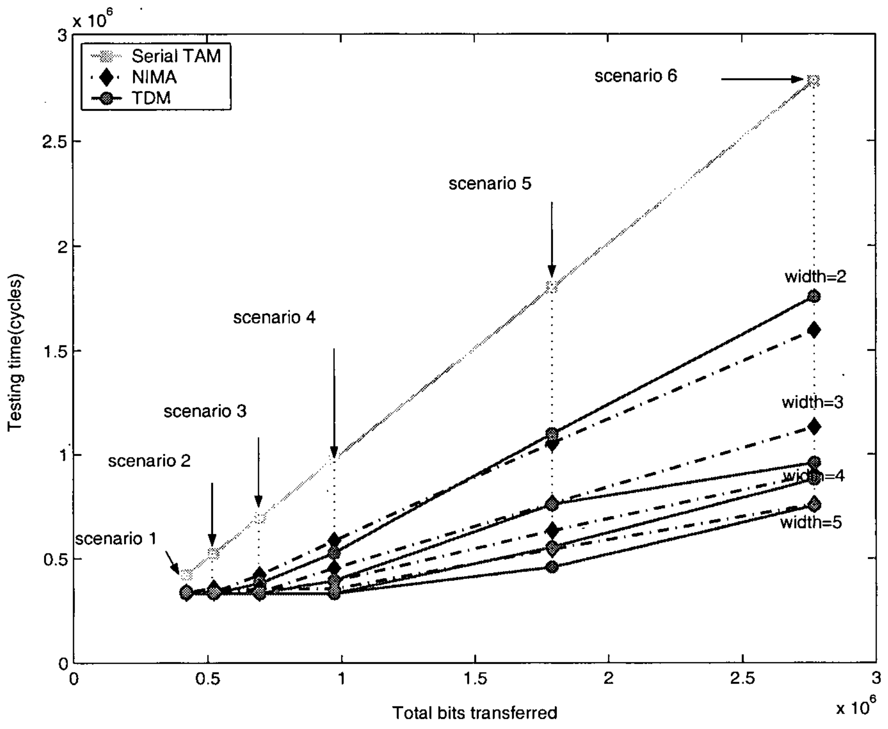


Figure 4.10: Test Time for Serial P1500, NIMA, and TDM-TAM.

sources after fabrication using *dynamic masking.* This feature is particularly attractive for addressing test requirements that cannot be anticipated before fabrication, for example in cases requiring increased diagnostics due to one or more faulty cores on an SoC. We implemented the TDM-TAM on a network processor engine design, and compared area and test times of TDM-TAM to other proposed TAMs. We illustrated how TDM-TAM offers an attractive alternative to other TAMs, because of its smaller area requirements and shorter test time.

Chapter 5

Conclusions

In this thesis the issues of SoC test integration are discussed, a novel wrapper design considering all factors of test cost is introduced, and a new TAM based on TDM is reported. Proposed wrapper design and TDM-TAM are analyzed both experimentally and theoretically and their high performance is proven.

DFT and test generation for SoC are becoming major concerns in the semiconductor industry because manufacturing test costs are emerging as a difficult challenge [5]. Also, since test issues should be considered during the design phase (not after), tools observing DFT are badly needed. The first step in SoC test integration is wrapper design. In Chapter 2, some of the previous work on wrapper design and optimization are reported. In Chapter 3, a new wrapper design that can be used as part of a SoC test integration tool is proposed and implemented. The proposed method is extensively described theoretically, and many examples are reported to demonstrate its efficiency. A comparison between our wrapper design and existing methods is conducted using experiments on the largest core in ITC'02 SoC test benchmarks. The results show up to a 2.7% reduction in test time, a 58% reduction in TAM width (test I/O pins) and a 77% reduction in required ATE memory for testing. These improvements for one core translate into a dramatic cost decrease, if our method is used for all cores of SoC.

The next step in SoC test integration is TAM design. As part of this research in this area TAM optimization for general bus-based configuration was investigated. This work are published in [19] and [20]. In [19] heuristics methods (GA) was applied to find an optimal test access configuration. Since this approach can handle both serial and parallel test data transportation to cores, a test time of 40% less than the previous leading method [14], which was much slower (with a run time of hours compared to seconds) was achieved. In [20] the system was improved to handle power consumption and place-and-route constraints.

Multi-frequency SoC testing is being performed by expensive ATEs and complicated ATE programming. In Chapter 4, a novel TAM, able to handle multi-frequency SoC testing, is proposed and implemented. Experiments show that for a nearly minimum possible test time, the overhead area of this TAM is very small (less than 2% as compared to 53% for NIMA [45]). An optimization method for this specific TAM is also reported.

5.1 Future Work

To complete a tool package for SoC test integration, SoC test scheduling should be studied and included in one package. Test scheduling is used after fabrication to determine in what order testing of the cores should be conducted. A test scheduling scheme takes the information related to TAM design and test data for each core as its input. Of the several methods available for scheduling, one should be selected and used to determine the best order of testing from the input data to minimize the test cost.

Also, a complete TAM design and optimization package should be developed. Using this package, the SoC integrator would have the option to choose the right type of TAM for the SoC. In this research, our focus was on bus-based TAM and TDM-TAM. A complete TAM design tool, however, should include different types of TAMs.

The last step in making the SoC test package is to integrate tools for wrapper design/optimization, TAM design/optimization and test scheduling) into a single package. So far, we have developed independent software tools for wrapper design and TAM design. After the test scheduling method (mentioned above) is complete, these three stand-alone techniques should be integrated into a single package, making a novel and much needed tool for SoC test design.

5.2 Contributions of this Work

This work has made the following contributions to the SoC test research community:

1. A novel wrapper design and optimization method has been introduced to minimize the core test time, the TAM width (translates to test I/O pins) and the required ATE memory. While previous methods for wrapper design only minimize one or two of these parameters, the proposed method considers all of these factors in minimization. Also, the performance of the proposed method is superior when compared to previous methods.

2. The optimization of a general bus-based TAM architecture has been studied [19, 20]. We considered the following issues in designing an optimal TAM with the minimum test time: assignment of cores to test buses, distribution of the total test width among multiple test buses, and estimation of TAM width required for an upper bound on testing time. The proposed method is implemented using a genetic algorithm. The system is able to design an optimum test access architecture incorporating system level constraints on power consumption and place-and-route (arising from the functional interconnections amongst the cores). This component is not reported in this thesis.

3. A new TAM based on time domain multiplexing is reported. TDM-TAM characteristics are accurately derived. An example showed that this TAM can be very useful in testing multi-frequency SoCs. Experiments were performed on different SoCs and the overhead area and test time was compared to serial and NIMA TAMs.

Bibliography

[1] *Three Methods for Determining Pareto-Optimal Solutions of Multiple-Objective Problems. Directions in Large-Scale Systems*, pages 117–138. New York, Plenum Press, 1975.

[2] ISCAS 85 and 89 web sites. http://www.cbl.ncsu.edu/CBL_Docs/Bench.html, 1989.

[3] IEEE P1500 Embedded Core Test: Wrapper Interface Port, Wrapper Instruction Register and Wrapper Bypass. http://grouper.ieee.org/groups/1500/aug00/wir.pdf, 2000.

[4] IEEE P1500 web site. http://grouper.ieee.org/groups/p1500/, 2001.

[5] International Technology Roadmap for Semiconductors. http://public.itrs.net/Files/2001ITRS/Home.htm, 2001.

[6] National Institute of Standards and Technology. http://www.nist.gov/dads/HTML/nphard.html, 2001.

[7] ITC'02 SoC Test Benchmarks. http://www.extra.research.philips.com /itc02socbenchm/, 2002.

[8] J. Aerts and E.J. Marinissen. Scan chain design for test time reduction in core-based ICs. In *Proc. of International Test Conference*, pages 448–457, 1998.

[9] G. Ausiello, P. Crescenzi, and G. Gambosi nand V. Kann ... [et al.]. *Complexity and approximation : combinatorial optimization problems and their approximability properties.* Springer, Berlin ; New York, 1999.

[10] J. Bedsole, R. Raina, A. Crouch, and M. S. Abadir. Very low cost testers: Opportunities and challenges. *IEEE Design and Test of Computers*, 18:60–69, Sept 2001.

[11] A. Borodin and R. El-Yaniv. *Online computation and competitive analysis.* Cambridge University Press, 1995.

[12] B. Bottoms. The third millennium's test dilemma. *IEEE Design and Test of Computers*, 15:7–11, Oct 1998.

[13] M. L. Bushnell and V. D. Agrawal. *Essentials of Electronic Testing for Digital, Memory and Mixed-Signal VLSI Circuits.* Kluwer Academic, 2000.

[14] K. Chakrabarty. Design of a system-on-chip test access architecture using integer linear programming. In *Proc. of VLSI Test Symposium*, pages 127–134, 2000.

[15] K. Chakrabarty. Design of system-on-a-chip test access architectures under place-and-route and power constraints. In *Proc. of International Design Automation Conference*, pages 432–437, 2000.

[16] K. Chakrabarty. Optimal test access architecture for system-on-a-chip. *DAES*, 6:26–49, Jan. 2001.

[17] A. Chandra and K. Chakrabarty. System-on-a-chip test data compression and decompression architectures based on golomb codes. *IEEE Transactions on Computer-Aided Design*, 20:113–120, March 2001.

[18] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein. *Introduction to algorithms.* MIT Pres, Cambridge, Mass., 2nd edition, 2001.

[19] Z. Ebadi and A. Ivanov. Design of an optimal test access architecture using genetic algorithm. In *Proc. of Asian Test Symposium*, pages 205–210, 2001.

[20] Z. Ebadi and A. Ivanov. Design of an optimal test access architecture under power and place-and-route constraints using ga. In *Proc. of Latin-American Test Workshop*, pages 154–159, 2002.

[21] Y. Zorian E.J. Marinissen. Challenges in testing core-based system ics. *IEEE Communications Magazine*, 37(6):104–109, June 1999.

[22] I. Ghosh, N.K. Jha, and S. Dey. A low overhead design for testability and test generation technique for core-based systems. In *Proc. of International Test Conference*, pages 50–59, 1999.

[23] S.K. Goel and E.J. Marinissen. TAM architecture and their implication on test application time. In *Proc. of International workshop on Test Embedded Core-based Systems*, pages 3.3–1–10, 2001.

[24] P.T. Gonciari, B.M. Al-Hashimi, and N. Nicolici. Useless memory allocation in system-on-a-chip test: problems and solutions. In *Proc. of VLSI Test Symposium*, pages 423–429, 2002.

[25] R.L. Graham. Bounds on multiprocessing anomalies. *SIAM Journal of Computing*, 17:416–429, 1969.

[26] I. Hamzaoglu and J.H. Patel. Test set compaction algorithms for combinational circuits. *IEEE Trans. on Computer Aided Design of Integrated Circuits and Systems*, 19:957–963, Aug. 2000.

[27] Dorit S. Hochbaum. Integer programming and combinatorial optimization: Lecture. Lecture notes of Course IEOR 269, Department of Industrial Engineering and Operations Research, University of California Berkeley.

[28] Dorit S. Hochbaum. *Approximation Algorithms for NP-Hard Problems.* PWS Publishers, 1 edition, January 1996.

[29] L. Hong, M. Nahvi, R. Fung, A. Ivanov, and R. Saleh. Novel test methodologies for soc/ip design implementation and comparison. In *Proc. of IEEE International Workshop on System-on-Chip for Real-Time Applications*, pages 20–30, 2002.

[30] V. Immaneni and S. Raman. Direct access test scheme-design of block and core cells for embedded ASICS . In *Proc. of International Test Conference*, pages 488–492, 1990.

[31] V. Iyengar and K. Chakrabarty. Test bus sizing for system-on-a-chip. *IEEE Transactions on Computers*, 51:229–459, May 2002.

[32] V. Iyengar, K. Chakrabarty, and E. J. Marinissen. Test wrapper and test access mechanism co-optimization for system-on-a-chip. *Journal of Electronic Testing: Theory and Applications (JETTA)*, 18:213–230, April 2001.

[33] V. Iyengar, K. Chakrabarty, and E. J. Marinissen. Test wrapper and test access mechanism co-optimization for system-on-a-chip. In *Proc. of International Test Conference*, pages 1023–1032, 2001.

[34] V. Iyengar, K. Chakrabarty, and E. J. Marinissen. Efficient wrapper/TAM co-optimization for large socs. In *Proc. of Design Automation and Test in Europe*, pages 491–498, 2002.

[35] V. Iyengar, K. Chakrabarty, and E. J. Marinissen. Integrated wrapper/TAM co-optimization, constraint-driven test scheduling, and tester data volume reduction for SOCs. In *Proc. of International Design Automation Conference*, pages 685–690, 2002.

[36] V. Iyengar, K. Chakrabarty, and E. J. Marinissen. On using rectangle packing for soc wrapper/tam co-optimization. In *Proc. of VLSI Test Symposium*, pages 253–258, 2002.

[37] A. Jas and N. Touba. Test vector decompression via cyclical scan chains and its application to testing core-based designs. In *Proc. of International Test Conference*, pages 458–464, 1998.

[38] B. Korte and J. Vygen. *combinatorial optimization, theory and algorithms.* Springer, 2000.

[39] J.H. Van Lint. *A course in combinatorics.* Cambridge University Press, 2nd edition, 2001.

[40] L.Whetsel. An IEEE 1149.1 based test access architecture for ic with embedded cores. In *Proc. of International Test Conference*, pages 69–78, 1997.

[41] E.J. Marinissen. A structures and scalable mechanism for test access to embedded reusable cores. In *Proc. of International Test Conference*, pages 284–293, 1998.

[42] E.J. Marinissen, S.K. Goel, and M. Lousberg. Wrapper design for embedded core test. In *Proc. of International Test Conference*, pages 911–920, 2000.

[43] E.J. Marinissen, V. Iyengar, and K. Chakrabarty. A set of benchmarks for modular testing of socs. In *Proc. of International Test Conference*, pages 519–528, 2002.

[44] E.J. Marinissen, Y.Zorian, R. Kapur, T. Taylor, and L. Whetsel. Towards a standard for embedded core test: an example. In *Proc. of International Test Conference*, pages 616–627, 1999.

[45] M. Nahvi and A. Ivanov. A packet switching communication-based test access mechanism for system chips. In *Proc. of European Test Workshop*, pages 81–86, 2001.

[46] N. Piersma. *combinatorial optimization and empirical process.* Thesis Publisher, 1993.

[47] M. Pinedo. *Scheduling: theory, algorithm and systems.* Prentice Hall, 1998.

[48] A. Sivaram. Split timing mode (stm) - answer to dual frequency domain testing. In *Proc. of International Test Conference*, pages 140–147, 2001.

[49] N.A. Touba and B. Pouya. Testing embedded cores using partial isolation rings. In *Proc. of International Test Conference*, pages 10–16, 1997.

[50] P. Varma and S. Bhatia. A structured test re-use methodology for core-based system chips. In *Proc. of International Test Conference*, pages 294–302, 1998.

[51] B. West and T. Napier. Sequencer per pin test system architecture. In *Proc. of International Test Conference*, pages 355–361, 1990.

[52] L. Whetsel. Core test connectivity communication and control. In *Proc. of International Test Conference*, pages 303–312, 1998.

[53] L. Whetsel. Addressable test ports, an approach to testing embedded cores. In *Proc. of International Test Conference*, pages 1055–1064, 1999.

[54] Y.Zorian, E.J. Marinissen, and S. Dey. Testing embedded-core based system chips. In *Proc. of International Test Conference*, pages 130–143, 1998.

Appendix A

P1500 Wrapper Elements

In this section we describe the P1500 wrapper [4] elements in detail.

A.1 Wrapper Boundary Cells

Wrapper Boundary Cells (see Figure A.1) are associated with the core terminals and provide controllability as well as observability for core terminals. These cells should support different modes:

- *Normal*: In this mode, the cell does not have any effect on the terminal, and the core functions normally.
- *Inward Facing*: In this mode, the test is directed toward the core, so it effects the core.
- *Outward Facing*: In this mode, the test is directed outward from the core. This mode is the mirror image of the Inward Facing mode.
- *Safe*: In this mode, the cell effects the core and ensures the wrapper does not damage core or system (a recommended mode).

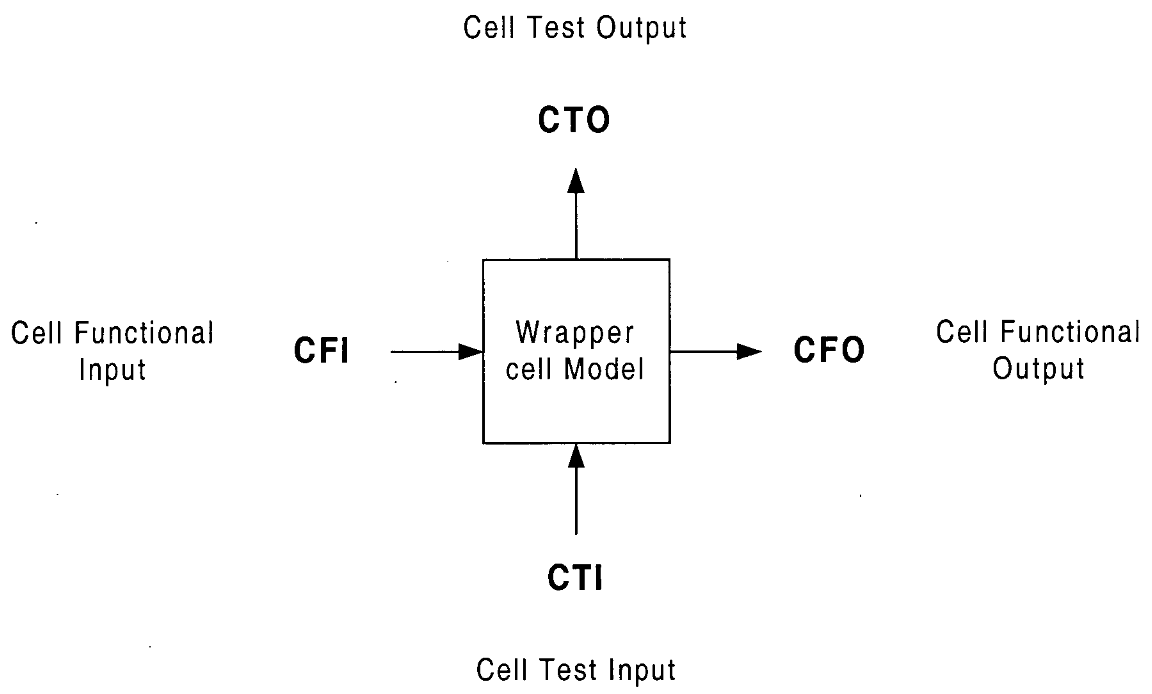


Figure A.1: Conceptual View of IEEE P1500 Wrapper Boundary Cell [4].

Also, the wrapper boundary cell event can be as follows:

- *Shift*: Move data through shift path
- *Capture*: sample data
- *Apply*: The moment when test data become active and effective
- *Update*: 1149.1-type update
- *Transfer*: Move data from Update element to shift path

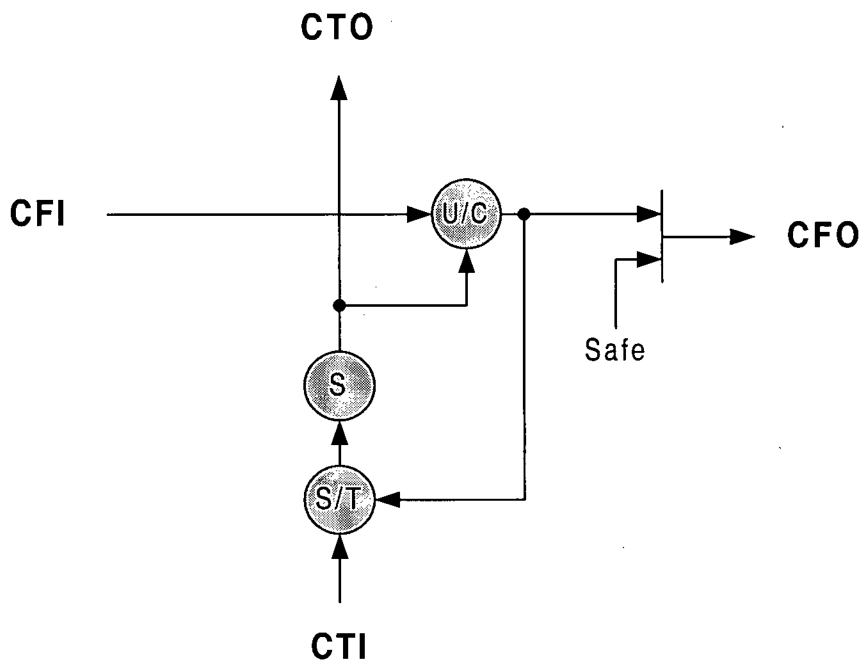


Figure A.2: P1500 Wrapper Boundary Cell: Cell Example Displaying all Events [4].

A.2 Wrapper Interface Port (WIP)

The WIP is defined to control and clock the Wrapper Instruction Register (WIR) and Bypass, Figure A.3. The WIP wrapper terminals include these:

- *WRCK* is one or more clocks used to operate registers.
- *WRSTN* is a dedicated asynchronous Wrapper Reset.
- *SelectWIR* selects whether or not the WIR is connected between WSI-WSO.

- *UpadteWR*,*ShiftWR* and *CaptureWR* are enables for register operations.

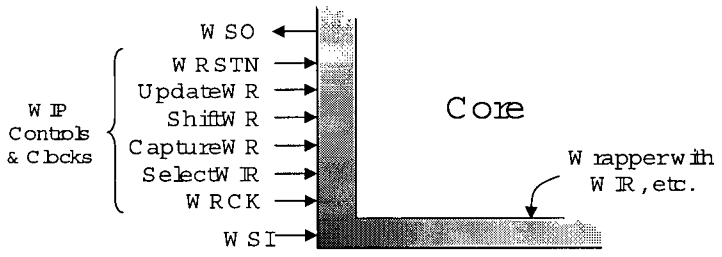


Figure A.3: P1500 Wrapper Interface Port (WIP) [3].

A.3 Wrapper Instruction Register (WIR)

The WIR is used to shift in and update instructions to the wrapper. The WIR circuitry (Figure A.4) is controlled and clocked by the Wrapper Interface Port (WIP) and provides the following:

- Serial shifts of the WIR contents from WSI to WSO
- Wrapper instruction decoding and circuitry updating
  - Generates Wrapper and Cores Modes based on Wrapper Instruction
  - Ensures that Modes remain stable during WIR shift operations
- Optional parallel capture of test control information into the WIR

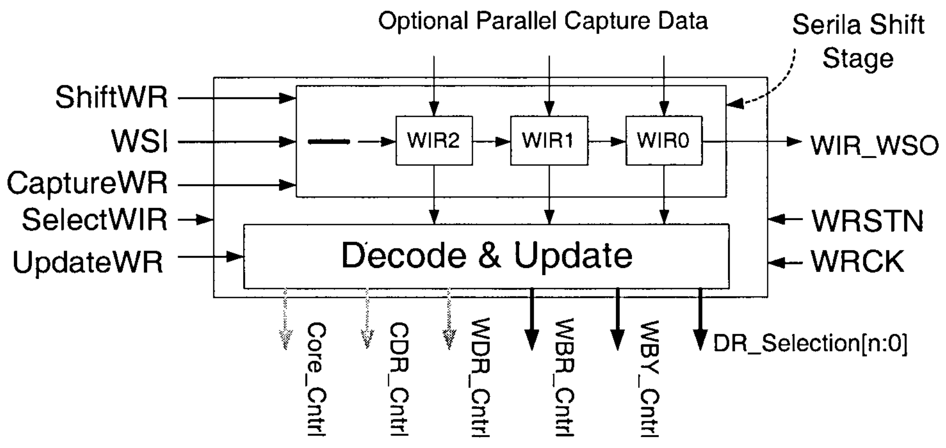


Figure A.4: P1500 Wrapper Instruction Register Circuitry [3].

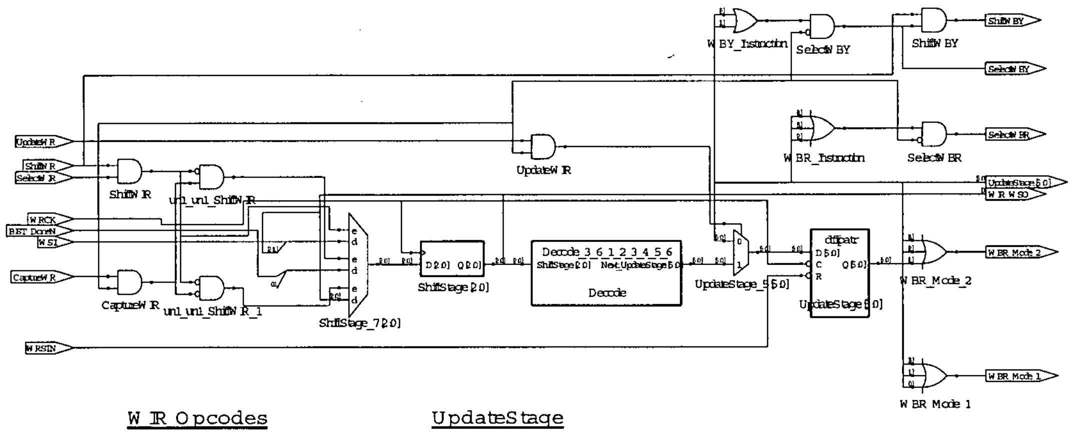
– can also be utilized for testing of WIR logic and WIR scan path

An example of WIR implementation is shown in Figure A.5.

A.4 Bypass Register

The Wrapper Bypass Register (WBY) provides a single bit scan bypass of Wrapper's SIL, from WSI to WSO (Figure A.6. The WBY is controlled and clocked by the WIP (e.g. , WRCK, WSI) and WIR Circuitry (e.g. WBY-Cntrl signals).

- The WBY Control signals from the WIR Circuitry are generated based on the current Wrapper Instruction and the WIP.
- WBY can only be selected when the WIP SelectWIR signal is logic 0.



W IR Opcodes	UpdateStage
W BYPASS_Opcode = 3 b001	W BYPASS = [5]
W PRELOAD_Opcode = 3 b010	W PRELOAD = [4]
W EXTEST_Opcode = 3 b011	W EXTEST = [3]
SAFESTATE_Opcode = 3 b100	SAFESTATE = [2]
W CLAMP_Opcode = 3 b101	W CLAMP = [1]
COREBIST_Opcode = 3 b110	COREBIST = [0]

Figure A.5: P1500 Wrapper Instruction Register: Example Implementation [3].

An example of WBR implementation is shown in Figure A.7.

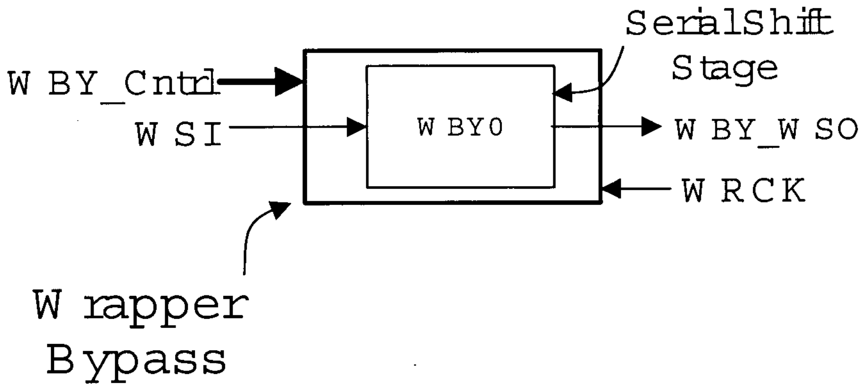


Figure A.6: P1500 Wrapper Bypass Register [3].

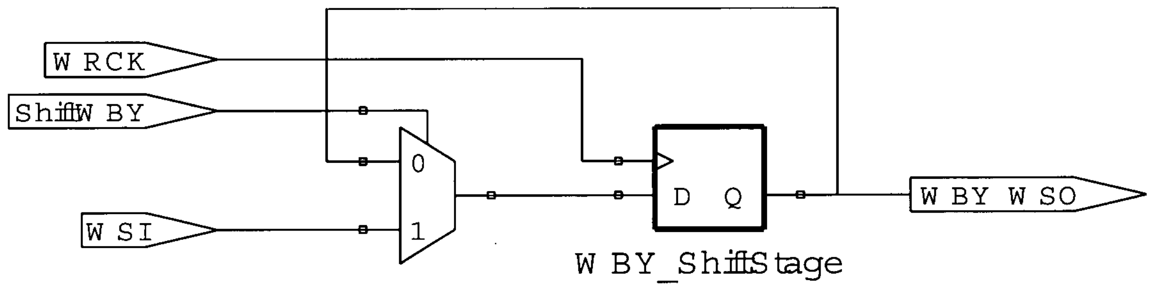


Figure A.7: P1500 Wrapper Bypass Register: Example Implementation [3].

Appendix B

Minimum Makespan Problem

B.1 Problem Formulation

Suppose that we have m identical machines an there are n jobs, each with a processing time of p_j, $j = 1 \dots n$, which must be processed on these machines. We are given that $n > m$ and no preemption of jobs is allowed, which means that each job must be processed on the same machine from start to finish. The idea is to partition the n jobs into m sets and create a schedule. Our objective in the schedule is to minimize the maximum machine makespan; that is, we minimize the latest finish time of the last job completed [28]. The decision version of this problem is as follows:

Is there a schedule (partition of n jobs into m subsets, S_i) such that

$$\textstyle\sum_{j \in S_i} p_j \leq K \qquad \forall S_i$$

This problem is known to be strongly NP-complete. A reduction from Bin-packing is possible.

B.1.1 Strongly NP-complete Problems

Definition The running time of an algorithm for a strongly NP-complete problem is independent of the size of the numbers in the input.

Alternatively, strongly NP-complete problems remain NP complete even if the input is given in unary. An NP-complete problem that is not strongly NP complete is called "weakly NP-complete." An example of a weakly NP complete problem is the 0/1 Knapsack problem, which has a running time of $\mathcal{O}(nB)$.

B.2 Heuristics for Solving the Problem

B.2.1 An Example using the LIST Heuristic

We illustrate the list heuristic with the following example. Suppose we have 6 machines, and the following list of jobs: 5 jobs of length 5, 5 jobs of length 1, and 1 job of length 6. The list heuristic works as follows:

List heuristic: Given a list of jobs, take each job one at time, and place it in the machine with the current earliest finish time.

Applying this rule results in a schedule with a makespan of 11, using the list as given above. The optimal schedule has a makespan of 6: put the job of length 6 on one machine, and the other five machines have one job of length 5 and one of length 1. So the list heuristic gives us a ratio of 11/6. Although not perfect, one advantage of the list heuristic is that it is an on-line algorithm. That is, it does not look ahead in the data before choosing an allocation. The following is a theorem [Graham 1966] providing a bound on the error of the list heuristic.

Theorem B.2.1 *The list heuristic is a $(2 - \frac{1}{m})$-approximation algorithm.*

Proof Let $\mathcal{OPT}$ be the optimal solution value, and $\mathcal{L}$ be the makespan returned by the LIST heuristic. First, observe that the best makespan that can be achieved with preemption is the sum of the job lengths, divided by m, the number of machines. The optimal solution must be greater than that solution obtained with preemption, and must also be greater than the longest job. Thus,

$$\mathcal{OPT} \geq \max\{\frac{1}{m}\sum_{j=1}^{n} p_j,\ \max_j p_j\}$$

Now, let t be the processing time of the last job on the list. Then,

$$\mathcal{L} - t \leq \frac{1}{m}(\sum_{j=1}^{n} p_j - t)$$

$$\Rightarrow m(\mathcal{L} - t) \leq m.\mathcal{OPT} - t$$

$$\Rightarrow \mathcal{L} \leq \frac{m.\mathcal{OPT} + (m-1)t}{m}$$

Since $t \leq \mathcal{OPT}$, then $\mathcal{L} \leq \frac{(2m-1).\mathcal{OPT}}{m} = (2 - \frac{1}{m})\mathcal{OPT}$. Thus, the heuristic is approximation $(2 - \frac{1}{m})$ algorithm. ∎

B.2.2 An Improvement: The LIST DECREASING Heuristic

To improve upon the list heuristic, we can sort the list in non-increasing order prior to assigning jobs to machines. This gives us the list decreasing heuristic.

List Decreasing heuristic: Sort list of jobs in non-increasing order, then apply the list heuristic.

The example in class had 6 machines, and a list of 13 jobs. The times for the jobs, in non-increasing order are 11, 11, 10, 9, 9, 8, 8, 7, 7, 6, 6, and 6. When we applied the heuristic, we obtained a makespan of 23. The optimal value is actually 18, so our ratio for the heuristic is 23/18. An alternate theorem by Graham states the quality of solution obtained with this heuristic.

Theorem B.2.2 *List Decreasing is a $(\frac{4}{3} - \frac{1}{3m})$-approximation.*

Proof We show that either this magnitude of error is valid, or the heuristic is optimal. Let us assume that the jobs are indexed so that $p_1 \geq p_2 \geq \cdots \geq p_n$.

Again, let $\mathcal{OPT}$ be the optimal makespan and $\mathcal{LD}$ be the makespan returned by the List Decreasing heuristic. If we suppose that the above theorem is false, this implies that $\frac{4}{3} - \frac{1}{3m} < \frac{\mathcal{LD}}{\mathcal{OPT}}$ for some problem instances. Since list decreasing is a list heuristic, theorem B.2.1 applies, and $\frac{\mathcal{LD}}{\mathcal{OPT}} \leq 1 + \frac{(m-1)t}{m.\mathcal{OPT}}$. For list decreasing, $t = p_n$, then

$$\frac{4}{3} - \frac{1}{3m} < 1 + \frac{(m-1)p_n}{m.\mathcal{OPT}}$$

$$\Rightarrow \frac{4m-1}{3} < m + \frac{(m-1)p_n}{\mathcal{OPT}}$$

$$\Rightarrow \frac{m-1}{3} < \frac{(m-1)p_n}{\mathcal{OPT}} \Rightarrow \mathcal{OPT} < 3p_n$$

$\mathcal{OPT} < 3p_n$ implies that at most two jobs are assigned to each machine, and $n < 2m$. If $n < 2m$, then add $2m - n$ jobs of length 0, therefore, there are $2m$ total jobs. Next, we claim that list decreasing is an optimal heuristic for $2m$ jobs. Consider this solution: simply assign the first m job, one to each machine, then assign the remaining m jobs to the machines, starting with the last machine assigned to a job in the first pass, and end with the first machine. We can show that this produces an optimal solution:

Compare the "pairing up" solution (described above) produced to an optimal solution. Suppose that these solutions differ first at the ith machine, where job i is matched to job $2m - i + 1$ in the "pairing up" solution, but matched with job j in the optimal solution. Likewise, machine k, $k > i$, is paired up with machine j, but paired up with $2m - i + 1$ in the optimal solution. Here is a summary:

Optimal		List Decreasing
$p_k + p_{2m-i+1}$	$\rightarrow$	$p_k + p_j$
$p_i + p_j$	$\rightarrow$	$p_i + p_{2m-i+1}$

Then,

$$p_k + p_j < p_i + p_j \quad \text{because} \quad p_k < p_i$$

$$p_i + p_{2m-i+1} < p_i + p_j \quad \text{because} \quad p_i < p_{2m-i+1}$$

If $p_i + p_j$ were the optimal makespan length, then the pairing up solution delivered by the heuristic would improve the solution, by the above inequalities. However, this would contradict our initial assumption that the optimal solution differs at the ith position. If $p_i + p_j$ is less than the optimal makespan length, then the heuristic has simply delivered a different optimal solution.

There may be instances where the list decreasing solution is different than the "pairing up" solution given above. However, we can show that the list decreasing solution is at least as good as "pairing up". For example, find a job i in the list decreasing assignment that is in a machine by itself (which means another machine contains three jobs). Now, we know that $\mathcal{LD} - p_{2m} \leq p_i$, because otherwise, job $2m$ would be on the same machine as job i. Using this inequality, and the fact that $p_{2m} \leq p_{2m-i+1}$, we have the following:

$$\mathcal{LD} \leq p_i + p_{2m} \leq p_i + p_{2m-i+1} \leq \text{Pairing Up Makespan}$$

Now, since list decreasing is at least as effective as "pairing up", then for $2m$ machines, the list decreasing heuristic is optimal. We have reached a contradiction, since we know the list decreasing does not always produce an optimal solution. So, this heuristic must be a $(\frac{4}{3} - \frac{1}{3m})$-approximation. ∎

A strong relationship exists between the MAKESPAN problem and the BIN-PACKING problem: suppose we knew the optimal makespan, and called it D. This quantity becomes the size of the bins, into which we must pack n items of varying sizes p_j. The number of bins used must be at most m, the number of available machines. If the value of D is unknown, it can first be guessed as the midpoint of a known interval. Then, if more than m bins were used, D must lie in the larger half of the interval. If less than m bins is used, then the smaller half of the interval must contain D. This procedure can be repeated, leading to a binary search approach.

From the above, the optimal solution to the BIN-PACKING problem leads to an optimal solution to the MINIMUM MAKESPAN problem. The complexity of this algorithm for the makespan problem is the number of calls times the complexity of the BIN-PACKING problem. How many calls are necessary? Consider this interval which contains the following:

$$\lfloor \max\{\frac{1}{m}\sum_{j=1}^{n} p_j, \max_j p_j\}, \frac{1}{m}\sum_{j=1}^{n}(p_j) + \min_j p_j \rfloor$$

The upper limit of this interval is an upper bound on the optimal solution, as it is achieved by the list heuristic. Therefore, the number of calls is at most logarithmic in the length of the above interval, and hence, polynomial. Unfortunately, BIN-PACKING is also known to be NP-complete. Thus, instead of solving it optimally, we solve it with a *dual approximation scheme.* If one uses a dual algorithm for linear programming, primal feasibility is often violated in choosing the solutions to the dual problem. Likewise, we "violate" feasibility by allowing our bins to be slightly larger than D. Basically, our dual approximation solution has two properties:

1. Solution is super-optimal. Number of bins used $\leq$ optimal number of bins.

2. There is an ϵ violation of feasibility: we allow bins of size $\leq (1+\epsilon)D$.

This approximation scheme is the subject of the next section, where the concept of scaling and grouping is introduced.

B.3 Polynomial Approximation Scheme to "Makespan"

Consider two similar problems: makespan and bin packing. In the makespan problem, we want to pack several objects of given sizes into a given number of bins, and to minimize the bins' maximum necessary capacity. The original formulation involved scheduling jobs of different lengths to run on a given number of machines so that the whole system finishes as early as possible (these formulations are equivalent). In the bin-packing problem, the number of bins is variable, but their capacities are constant. The two problems are dual to each other. Both are invariant to scaling, that is, feasibility, optimality and so forth do not depend on the units of measurement. Both are known to be strongly NP-hard. Suppose we have a super-optimal solution to the bin packing problem, an instance that is known to use at most as many bins as the optimal, but may "overhang" by a factor of ε, making it infeasible. This solution can then be used to approximate the corresponding makespan problem in the following way:

```
Let $L = \max\{\max_j t_j, \frac{1}{m}\sum_i t_i\}$.
Guess a median $d \in [L, 2L]$.
Find an $\varepsilon$-dual approximation for packing $m$ bins of size $d$.
If less than $m$ bins were used, search the smaller half of the interval.
If $m$ bins proved to be too few, search the larger half of the interval.
Repeat until length of the interval is less than 1.
```

Here, we look for a scheme to find ε-approximations to bin packing in time polynomial in n (number of items) for each given ε. Our algorithm runs as follows. (1) Pack all items larger than ε (assuming bin sizes are normalized to 1). And (2) pack the remaining items.

In stage 1, no more than $\frac{1}{\varepsilon}$ items can go into a single bin. The problem would be easier with a fixed number of possible sizes, because then dynamic programming could be used. Instead, we use a technique called *scaling* and *grouping.* Divide the interval $(0, \varepsilon)$ into subintervals of size ε^2 . Divide every item's size by $\frac{1}{\varepsilon}$, rounding down to the nearest multiple of ε^2. Then solve the problem optimally using dynamic programming. We round down to ensure super-optimality, since if item sizes are reduced or kept the same, fewer or as many bins are required to accommodate them than before.

A feasible configuration is a collection of items that can feasibly go into one bin. There are $s = \frac{1}{\varepsilon^2}$ possible sizes, and for each size, at most $\frac{1}{\varepsilon}$ can