NAIST-IS-DD0561211

Doctoral Dissertation

Studies on Core-Based Testing of System-on-Chips Using Functional Bus and Network-on-Chip Interconnects

Fawnizu Azmadi Hussin

September 18, 2008

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

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

Fawnizu Azmadi Hussin

Thesis Committee:

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

Studies on Core-Based Testing of System-on-Chips Using Functional Bus and Network-on-Chip Interconnects*

Fawnizu Azmadi Hussin

Abstract

The tests of a complex system such as a microprocessor-based system-on-chip (SoC) or a network-on-chip (NoC) are difficult and expensive. In this thesis, we propose three core-based test methods that reuse the existing functional interconnects–a flat bus, hierarchical buses of multiprocessor SoC's (MPSoC), and a NoC–in order to avoid the silicon area cost of a dedicated test access mechanism (TAM). However, the use of functional interconnects as functional TAM's introduces several new problems.

During tests, the interconnects–including the bus arbitrator, the bus bridges, and the NoC routers–operate in the functional mode to transport the test stimuli and responses, while the core under tests (CUT) operate in the test mode. Second, the test data is transported to the CUT through the functional bus, and not directly to the test port. Therefore, special core test wrappers that can provide the necessary control signals required by the different functional interconnect are proposed. We developed two types of wrappers, one buffer-based wrapper for the bus-based systems and another pair of complementary wrappers for the NoC-based systems.

Using the core test wrappers, we propose test scheduling schemes for the three functionally different types of interconnects. The test scheduling scheme for a flat bus is developed based on an efficient packet scheduling scheme that minimizes both the buffer sizes and the test time under a power constraint. The scheduling

* Doctoral Dissertation, Department of Information Processing, Graduate School of Information Science, Nara Institute of Science and Technology, NAIST-IS-DD0561211, September 18, 2008.

scheme is then extended to take advantage of the hierarchical bus architecture of the MPSoC systems. The third test scheduling scheme based on the bandwidth sharing is developed specifically for the NoC-based systems. The test scheduling is performed under the objective of co-optimizing the wrapper area cost and the resulting test application time using the two complementary NoC wrappers.

For each of the proposed methodology for the three types of SoC architecture, we conducted a thorough experimental evaluation in order to verify their effectiveness compared to other methods.

Keywords:

Wrapper design, functional TAM, SoC test scheduling, NoC-reuse

List of Publications

Journal Paper

1. Fawnizu Azmadi Hussin, Tomokazu Yoneda, Alex Orailoglu and Hideo Fujiwara, "Scheduling power-constrained tests through the SoC functional bus," IEICE Transactions on Information and Systems, Vol. E91-D, No. 3, pp. 736–746, Mar. 2008.

2. Fawnizu Azmadi Hussin, Tomokazu Yoneda and Hideo Fujiwara, "NoC-compatible wrapper design and optimization under channel bandwidth and test time constraints," IEICE Transactions on Information and Systems, Vol. E91-D, No. 7, pp. 2008–2017, July 2008.

3. Fawnizu Azmadi Hussin, Tomokazu Yoneda and Hideo Fujiwara, "On NoC bandwidth sharing for the optimization of area cost and test application time," IEICE Transactions on Information and Systems, Vol. E91-D, No. 7, pp. 1999–2007, July 2008.

International Conferences (Reviewed)

1. Fawnizu Azmadi Hussin, Tomokazu Yoneda, Alex Orailoglu and Hideo Fujiwara, "Power-constrained SoC test schedules through utilization of functional buses," 24th IEEE International Conference on Computer Design (ICCD'06), pp. 230–236, Oct. 2006.

2. Fawnizu Azmadi Hussin, Tomokazu Yoneda, Alex Orailoglu and Hideo Fujiwara, "Core-based testing of multiprocessor system-on-chips utilizing hier-

archical functional buses," 12th Asia and South Pacific Design Automation Conference 2007 (ASP-DAC'07), pp. 720–725, Jan. 2007.

3. Fawnizu Azmadi Hussin, Tomokazu Yoneda and Hideo Fujiwara, "Optimization of NoC wrapper design under bandwidth and test time constraints," 12th IEEE European Test Symposium (ETS'07), pp. 35–40, May 2007.

4. Fawnizu Azmadi Hussin, Tomokazu Yoneda and Hideo Fujiwara, "Area overhead and test time co-optimization through NoC bandwidth sharing," IEEE 16th Asian Test Symposium (ATS'07), pp. 459–462, Oct. 2007.

Technical Reports

1. Fawnizu Azmadi Hussin, Tomokazu Yoneda, Alex Orailoglu and Hideo Fujiwara, "Power-conscious microprocessor-based testing of system-on-chip," Technical Report of IEICE (VLD2006-6), Vol. 106, No. 32, pp. 25–30, May 2006.

2. Fawnizu Azmadi Hussin, Tomokazu Yoneda and Hideo Fujiwara, "NoC wrapper optimization under channel bandwidth and test time constraints," Technical Report of IEICE (DC2006-80), Vol. 106, No. 528, pp. 1–6, Feb. 2007.

Award

1. IEEE Kansai Section Student Paper Award, Feb. 2008.

For my wife, who offered me unconditional love and support throughout the course of this thesis.

Fazleen Mustapha

And to my son, may you be inspired.

Faheem Fawnizu

"We choose to go to the moon."

"We choose to go to the moon in this decade and do the other things, not because they are easy, but because they are hard, because that goal will serve to organize and measure the best of our energies and skills, because that challenge is one that we are willing to accept, one we are unwilling to postpone, and one which we intend to win, and the others, too."

–John Fitzgerald Kennedy, September 12, 1962

Some things are worth doing not because they are easy, but because they are hard.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 The Trend Toward a Core-based Test Methodology

Tests of an integrated circuit are required in order to weed out the bad chips from the good ones. Chip test is an expensive process in the manufacturing of an integrated circuit. From the design perspective, in order to keep up with the Moore's Law in terms of design complexity and the transistor counts, chip designers break the designs into independent subsystem blocks (commonly known as intellectual property (IP) cores). Each subsystem can be a simple combinational logic or can be as complex as a multi-core microprocessor. Such design style is called a System-on-a-Chip (SoC).

System designers are adapting to the SoC design methodology because of its efficiency compared to the traditional system-on-board approach. The main benefit of the SoC approach is that it can drastically shorten the design cycle by allowing pre-designed cores and their associated test sets to be reused. The International Technology Roadmap for Semiconductors (ITRS) 2007 Edition [2] describes the *Increasing Device Integration* trends as one of the Key Drivers in the Test and Test Equipment section.

> *"... Increased device integration forces a re-integration of test solutions to maintain scaling of test costs and product quality. The op-*

timized test solutions for stand-alone RAMs, cores, and other blocks typically do not scale linearly without modification, additional DFT, or new partitioning to the integrated device test solutions. In particular, additional DFT in-die or even in-package may be required to provide access to and testing of embedded blocks and cores..."
—ITRS, 2007 Edition

Most of the industrial practitioners today does not yet embrace the core-based test approach as mentioned and recommended in the ITRS report. The preferred solution is still to flatten the core-based hierarchical design and apply the traditional test scheme—apply ATPG and test of the SoC as a single, flat, and extremely large and complex design. However, it makes more sense to consider core-based testing as the design complexity increases. This is particularly true when some of the embedded IP cores are outsourced from external IP providers. For that purpose, the IEEE 1500 standard [3, 4] was developed to support this core-based test reuse methodology.

Before discussing further the test methodology proposed in this dissertation, we will first look at the target SoC architecture and the test approach for the individual embedded logic core.

1.2 System-on-Chip Architecture

We can find several slightly different definitions of a system-on-a-chip. It is loosely defined as an entire system integrated on a single chip. It may include one or more cores with user-defined logic (UDL) integrated by the core user or system integrator. The general idea is that a SoC technology includes the design, manufacturing and packaging of all the necessary electronic circuits and parts for a complete "system" (such as the chips for cell phones or digital cameras) on a single integrated circuit. In the previous generation system-on-board (SoB), each functional module is designed, manufactured, packaged, and tested as an independent component or chip. Each fault-free packaged chip is soldered to the board during the board assembly stage to make a complete system.

Compared to the SoC, the SoB form factor is much larger. Furthermore, SoB design cannot take advantage of the shorter design cycle and higher cost efficiency of the SoC. The power dissipation as well as the inter-chip interconnect delay is much larger, thereby affecting the overall SoB's system performance. The inter-chip delay, which goes through the die-to-package wire-bonding and the I/O pad, is much larger compared to the inter-core delay (through the metal layers) of the SoC.

Different from that of SoC's, the tests of the individual chip in the SoB is performed by the chip manufacturer, and not the system integrator. The system integrator obtains each packaged chip (either from an external manufacturer or a separate design department from the same company) after it is tested and certified fault-free. Therefore, the system integrator does not have to worry about individual chip tests.

With SoC design technique, the whole SoB design can now be integrated onto a single chip. Even the passive components (capacitors and resistors) can be fabricated onto the same die. Instead of using a fabricated and tested chip as one of the modules, a system integrator uses a pre-fabricated module as part of the design. The module can either be obtained (purchased or licensed) from an outside source as an intellectual property (IP) core or designed separately by the same design group or different design group within the same company. The important characteristic of the SoC is the fact that the system is built from multiple smaller subsystems, which can be designed independently.

Each subsystem/module/core can be in various forms. Depending the format, the test engineer has different levels of freedom when dealing with the tests of the complete system. The following describes the three most common forms of an IP core.

- *Soft core* is a predesigned block of functional logic such as a macro, megacell, or memory with a register transfer level (RTL) representation. Soft cores are inherently process technology independent. With soft cores, the system integrator has the most freedom, both in terms of design integration and test.
- *Firm core* is a predesigned block of functional logic such as a macro, megacell, or memory that has a process technology-dependent netlist representation and may be amenable to some modification. It is typically in the

form of a synthesized gate-level netlist.

- *Hard core* is a predesigned block of functional logic such as a macro, megacell, or memory that has a physical implementation that cannot be modified, in the form of a completed layout. In this case, the IP provider is responsible to insert the design-for-testability (DfT) circuits into the core design (as shown by the three parallel core design flow in Figure 1.1); this includes the tasks of forming the scan chains and generating the test environment (test vectors/data and test configurations), and to provide the test environment for the IP core to the system integrator. In this case, the system integrator has no choice but to use the test information from the IP provider.

Figure 1.1 shows the design flow for a complete SoC system. The test flow included into the design flow assumes a dedicated TAM based architecture. After the system design, verification, and synthesis, the test access mechanism and core wrappers are added to the synthesized netlist. The system-level core information is used during this process. Depending on the methodology, the test scheduling step is performed concurrently with the TAM and wrapper design, or after its completion.

Figure 1.2 illustrates a complex system-on-a-chip which consists of an embedded ARM processor, memory (SRAM and Flash), and a host of supporting logic blocks. In addition, all these subsystems are connected together by a system bus (ASB/AHB) and a peripheral bus (APB). Together, they make up a complete functional system.

To test this hierarchical system, one way of doing it is by synthesizing the whole system to get a flat and large gate-level netlist. This is possible when the cores are either soft or firm formats. A hard core is not flexible and cannot be flattened with the rest of the system. After synthesis, the netlist is fed to a scan insertion tool which will convert all flip-flops into scan flip-flops, assuming a full-scan architecture is desired. The scan flip-flops are connected together into one or more scan chains. From here, the flip-flops (or scan flip-flops) can be isolated from the combinational logic circuits. The process continues by applying an automatic test pattern generation (ATPG) tool to the combinational parts of the design. These generated test patterns are then applied to the circuit using a boundary scan mechanism.

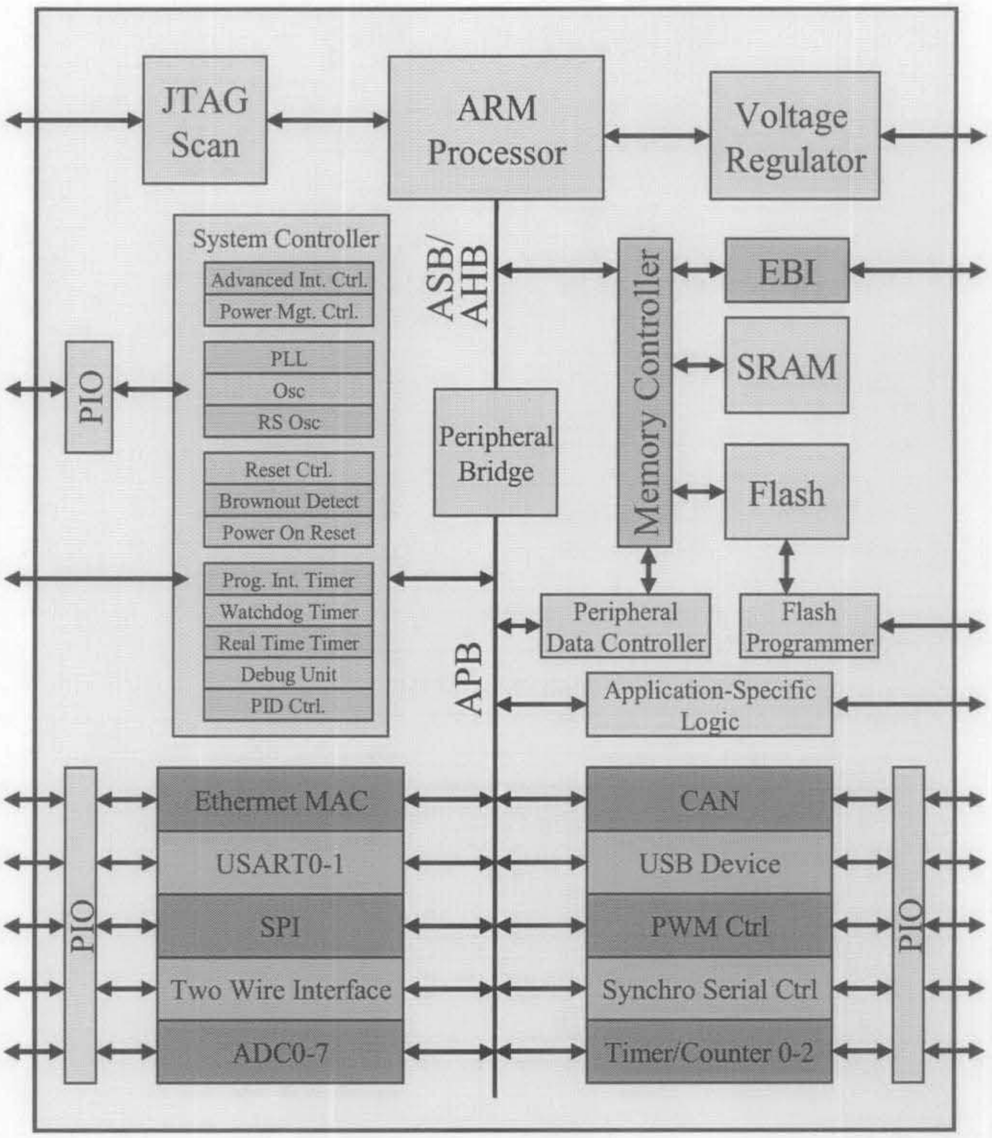


Figure 1.2: A microprocessor-based system-on-a-chip based on the ARM bus architecture.

- The simultaneous test application of the whole circuit might cause an excessively large test power dissipation. The test power is normally larger than the power dissipation during the intended normal operations
- It is common for a system to contain multiple instances of an IP core such

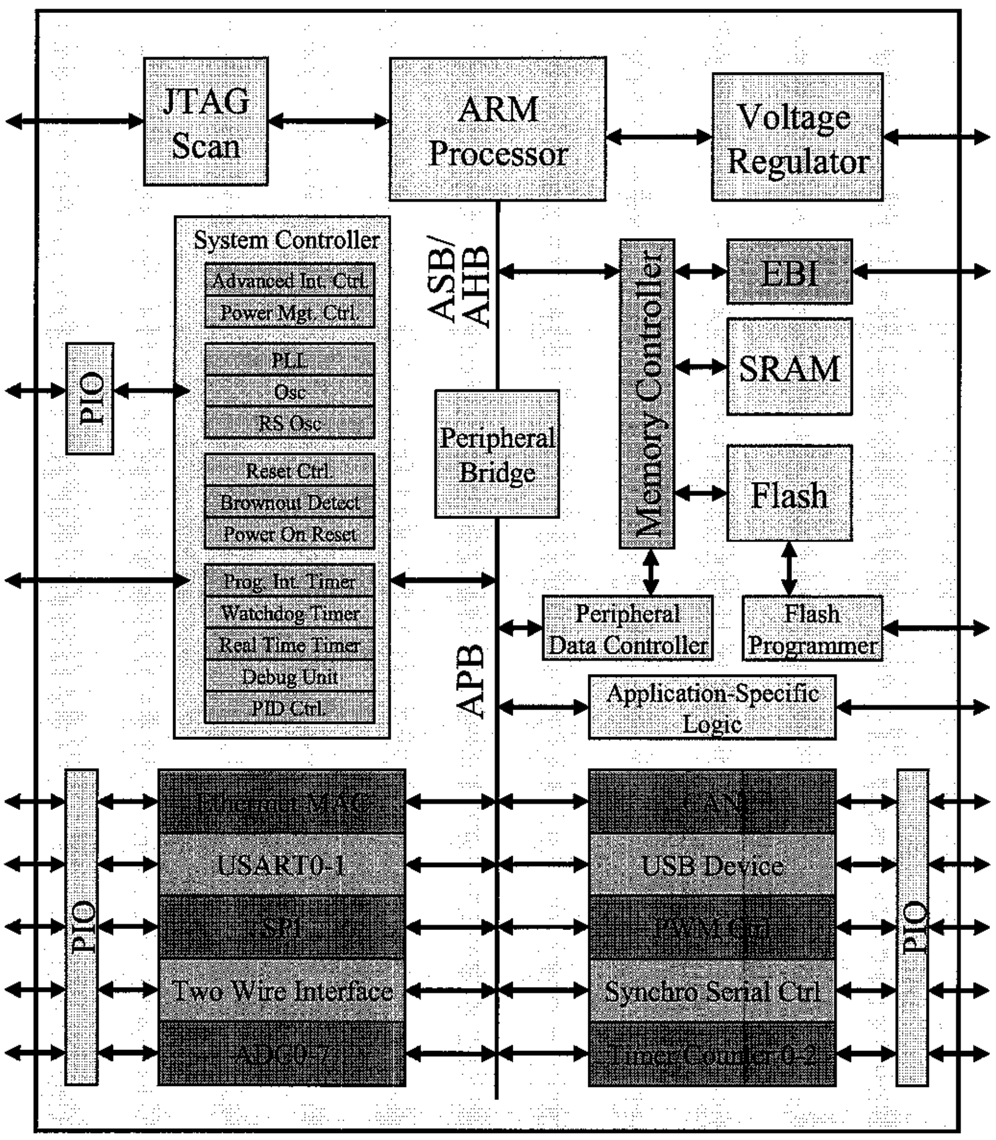


Figure 1.2: A microprocessor-based system-on-a-chip based on the ARM bus architecture.

- The simultaneous test application of the whole circuit might cause an excessively large test power dissipation. The test power is normally larger than the power dissipation during the intended normal operations
- It is common for a system to contain multiple instances of an IP core such

as memory and peripheral controllers. This test approach cannot take advantage of the duplicity in terms of

- the potential of reducing the storage requirement in the external automatic test equipment (ATE) by utilizing identical sets of the test stimuli and responses.
- the potential of reducing the test generation time because the same test data can be used for multiple identical IP cores. Some IP cores do not require this process if the IP providers include the test environment as part of the IP cores. Test environment provided by the IP provider is typically of higher qualify because they have a more intimate knowledge of the functionalities of the IP cores.
- the potential of shortening the test application time by simultaneously scanning in the test data in parallel for multiple identical IP cores.

For many years, the academic and industrial research community have been working on the core-based test application strategy for the SoC. In the following chapters, we will review some of the existing SoC test strategies and elaborate our proposed strategies to take advantage of the hierarchical nature of the SoC. In this chapter, we begin by laying the foundation by looking at the scan based test strategy for each individual IP core

1.3 Scan-based Testing of Embedded Cores

Figure 1.3 shows a simplified representation of a SoC with twelve cores without the interconnects. Like Figure 1.2, the IP cores are functionally connected to each other through a system of interconnects such as a flat or hierarchical bus. However, because the conventional test scheme that will be analyzed subsequently does not use the interconnects as part of its test architecture, we omit the interconnects from the SoC representation for clarity.

Definition 1.1 A *test access mechanism* (TAM) is a feature of an SoC design that enables the delivery of test data to and from cores or core wrappers. It is a mechanism for moving stimuli to and observing responses from a core or user-defined logic (UDL) during nonfunctional or test mode. The signals may

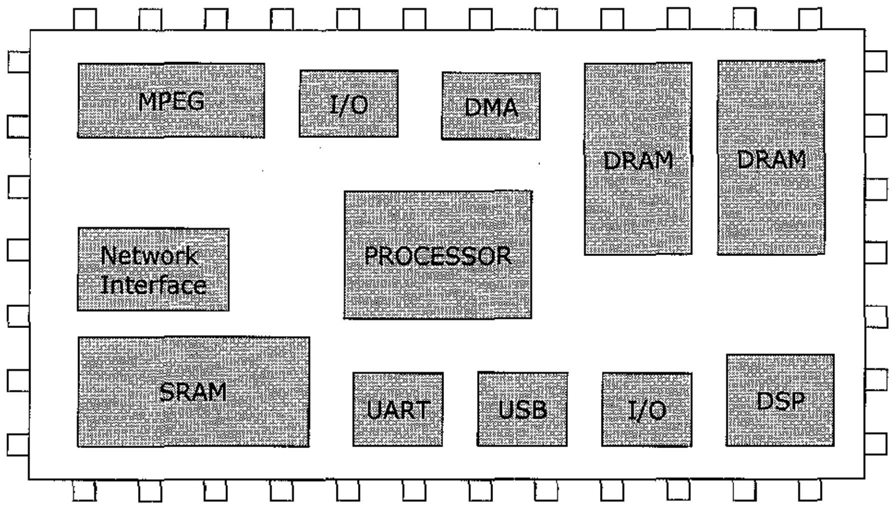


Figure 1.3: A simplified representation of the core-based SoC architecture without the functional bus interfaces.

be propagated to and from a core, from either embedded circuitry or from the primary inputs and outputs of the system chip [4] (Figure 1.4). ∎

In order to enable core-based testing, a network of these test access mechanisms (TAM) are added to the system (Figure 1.5). A TAM consists of two parts. One part for the actual test data transportation and one part that control the test data transportation. The TAM wires provide a direct pathway between an ATE and the IP core under test (CUT). The test vectors are transported to the CUT from the ATE input on the left side, applied to the CUT to obtain the test responses, then the test responses are transported out of the chip through the TAM wires on the right side back to the ATE for comparison with the expected test responses. In the example, the memory blocks (SRAM, DRAM) are ignored since they are normally and more effectively tested using dedicated memory test techniques such as memory built-in-self-test (memory BIST). Therefore, the tests of memory blocks are not considered.

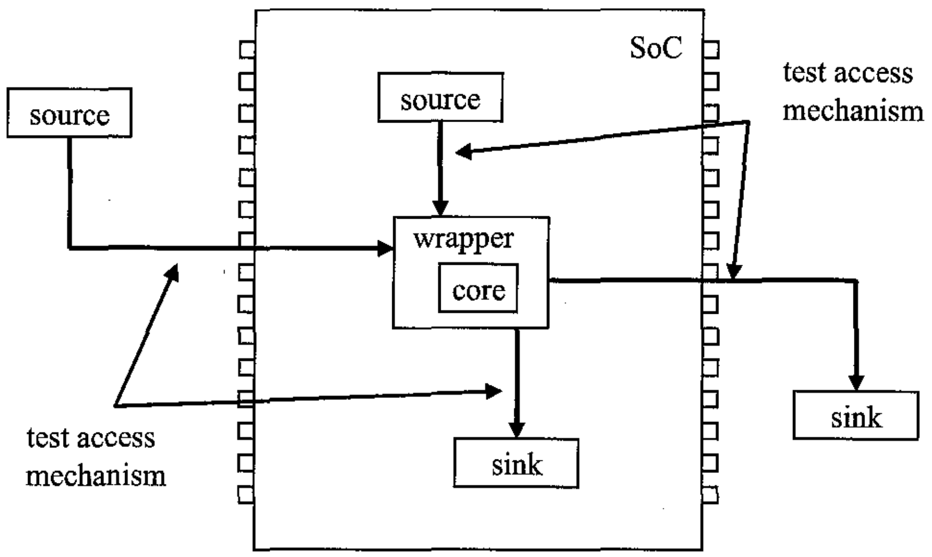


Figure 1.4: Test source and sinks interfaced to the core by means of a TAM and a wrapper. Illustration based on [5], reproduced with permission.

1.3.1 Core Test Architecture

Definition 1.2 A core *wrapper* is a circuitry added around an embedded core to facilitate test reuse and to interface between a test access mechanism (TAM) and the embedded core [4] (Figure 1.4). Test reuse is the ability to apply a predetermined test pattern associated with a core after this core has been integrated into an SoC. Test reuse is a consequence of design reuse and often requires the adaptation of a test protocol to reflect the core's new environment within an SoC. The use of core wrappers allow these test patterns to be reused in the environment of an SoC. ■

Figure 1.6 illustrates a Core A wrapped in the IEEE 1500 standard wrapper. The wrapper consists of several components. The wrapper boundary registers (WBR) act as input and output buffers. In Figure 1.6, five input wrapper boundary registers are located between the wrapper boundary inputs (d[0]-d[4] labels on the outside) and the Core A's primary inputs (d[0]-d[4] labels inside Core A). Similarly, three WBR's are located at the outputs (q[0]-q[2]).

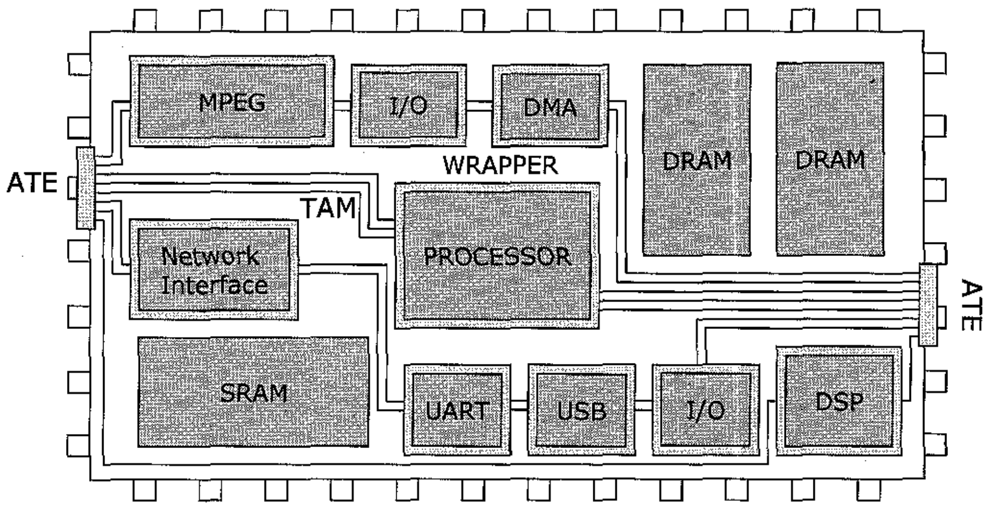


Figure 1.5: The SoC in Figure 1.3 after the core-based test architecture is inserted. The test access mechanism (TAM) provides external access to the embedded cores and the core wrappers provide test control and access to each individual core.

The second component is the wrapper instruction register (WIR), which accepts several predefined instructions from the WSC input. These instructions are used to set the wrapper modes. In the normal mode, the WBR selects the functional inputs (d[0]-d[4]) and outputs (q[0]-q[2]), respectively. In the test mode, the WBR selects the scan inputs (WSI input in Figure 1.6). Figure 1.6 shows the serial INTEST mode, where the wrapper is configured with one long scan chain to provide test stimuli (from WSI input) and to capture test responses (from WSO output) for Core A, hence the name INTEST (or internal test) mode.

Other test modes are parallel test mode (using the WPI and WPO), which can provide wider TAM access for shorter test application time, and bypass mode (using the WBY register), which is used when multiple cores are sharing the same TAM wires as shown in Figure 1.5. By setting bypass mode for all cores except one, each core can be tested sequentially using the shared TAM wires. This is used when limited number of TAM wires are available. In addition, the wrapper serial input (WSI) is typically used to connect serially all the cores in the SoC using a single wire. The WSI input is used to update the wrapper instruction register

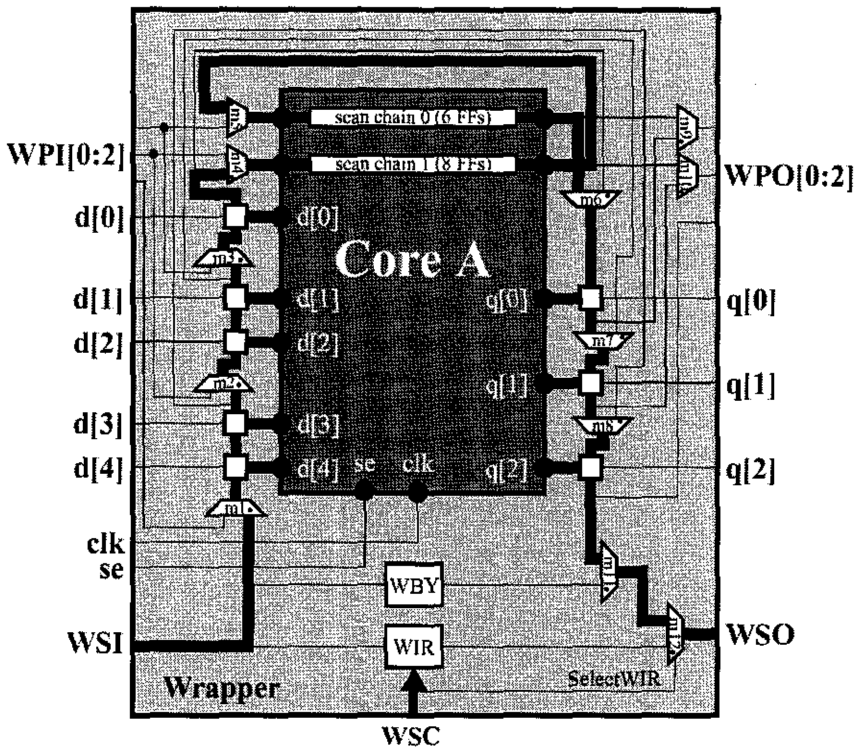


Figure 1.6: An IP core wrapper in the 1500 standard wrapper [4]. The wrapper is configured with the serial INTEST mode.

(WIR) of the target core. In the bypass mode, the bypass register eliminates all the boundary registers from the serial scan path to shorten the time to update the WIR.

Using the example in Figure 1.6, we can extract the single wrapper scan chain structure as shown in Figure 1.7. The wrapper scan chain consists of five WBR for the primary inputs $d[4:0]$, $8+6=14$ scan cells from the two 8- and 6-bit-long internal scan chains, and three WBR for the primary outputs $q[2:0]$. All together, the scan chain is 22 bits long. For this wrapper configuration, a pair of test vector and response of 19-bit and 17-bit long, respectively, are required. In other words, the scan-in depth (si_0) and the scan-out depth (so_0) for the wrapper configuration are 19-bits and 17-bits respectively. This concept is explained next.

The tests of an IP core can be broken down into three stages.

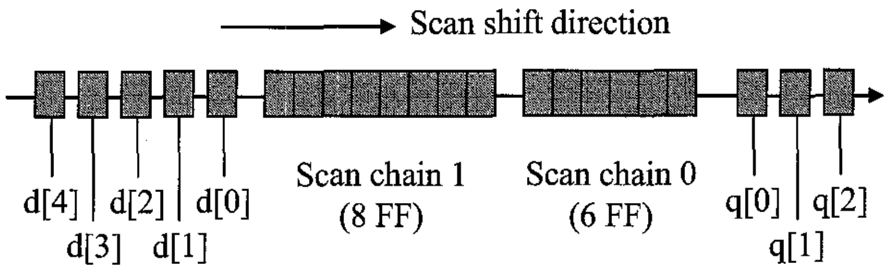


Figure 1.7: The single scan chain architecture from the serial intest wrapper configuration of Figure 1.6.

1. Stage 1 [Scan-in stage]: The test stimuli are loaded (or scanned-in) to the primary input WBR's (5 bits) and the internal scan chains (14 bits). The primary output WBR's do not require test stimuli because they are on the output side of the circuit under test (CUT), the combinational logic in Figure 1.8.
2. Stage 2 [Capture stage]: The test stimuli are applied to the CUT by applying a clock cycle. After this stage, the contents of the internal scan chains and the primary outputs WBR's, are updated to reflect the next state of the CUT, which is also called the test response. The contents of input WBR's are unchanged.
3. Stage 3 [Scan-out stage]: The test response, which consists of the contents of the internal scan chains (14 bits) and the primary output WBR's (3 bits) are unloaded (or scanned-out) of the scan chain to be compared to the expected response. If the circuit response and the expected response match for every test stimuli, the circuit is considered fault-free.

The total test application time for the single test stimuli is $19 + 1 + 17 = 37$ clock cycles. If the test set consists of two test stimuli v_0 and v_1 (where v_0 is applied before v_1, and the number of test stimuli is $n_v = 2$) , the scan-in stage of v_1 can be overlapped with the scan-out stage of v_0. Since the scan-in depth is larger than the scan-out depth by two bits in this example, the extra 2-bits scanned-out (at the output of the scan chain) during the scan-in of v_1 are ignored. On the other hand, if the scan-in depth is shorter than the scan-out depth by n

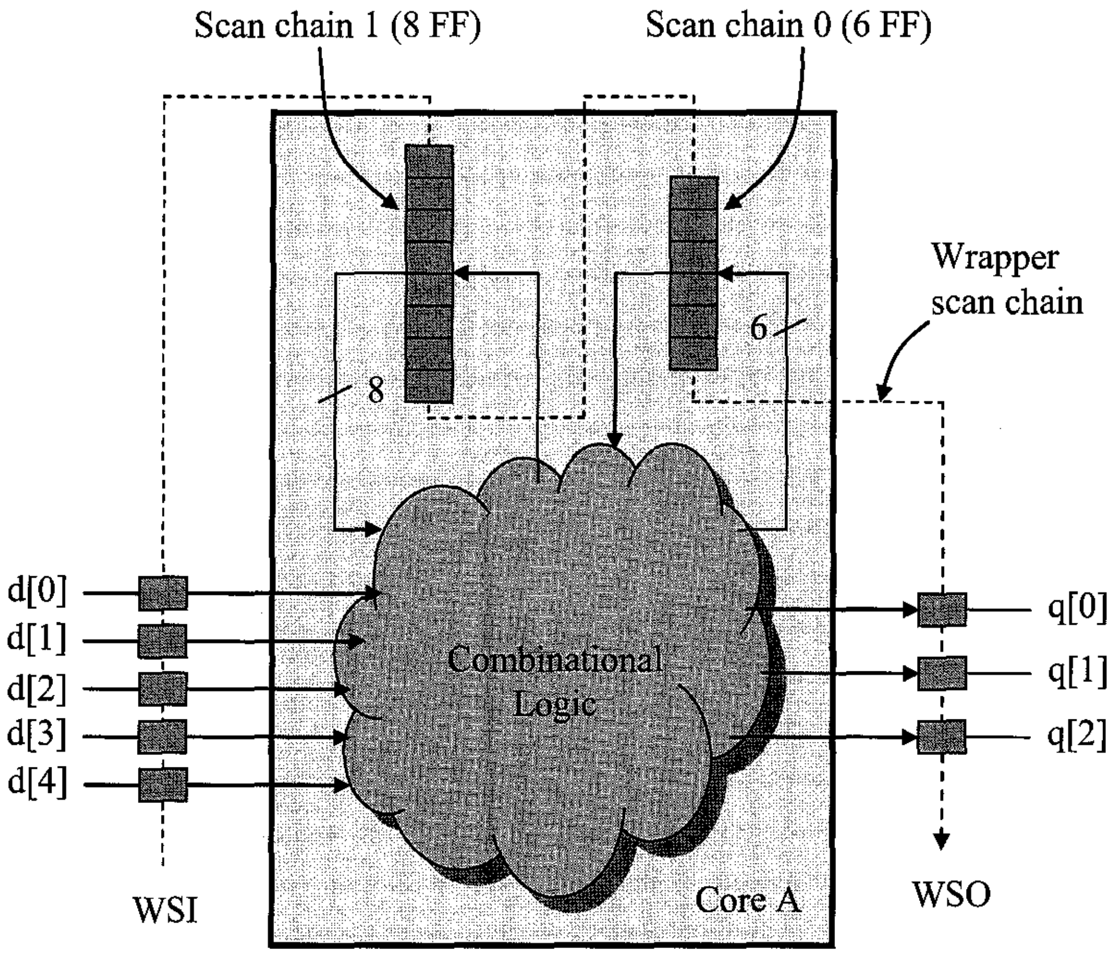


Figure 1.8: Alternative representation of the 1500-wrapped core in Figure 1.6. The IP core is divided into scan flip-flops and combinational logic components.

bits, the second test stimuli must be reformatted to accommodate the artificially elongated scan-in cycle (or simply scan cycle), by padding it with n dummy bits during the first n scan cycles. When the scan-in and scan-out stages are overlapped, the number of scan cycles follow the larger of the two. As a result, the test application time (TAT) for n_v test stimuli/response pairs for a single scan chain architecture in Figure 1.7 can be written as

$$TAT = (max\{si_0, so_0\} + 1) \times n_v + min\{si_0, so_0\} \tag{1.1}$$

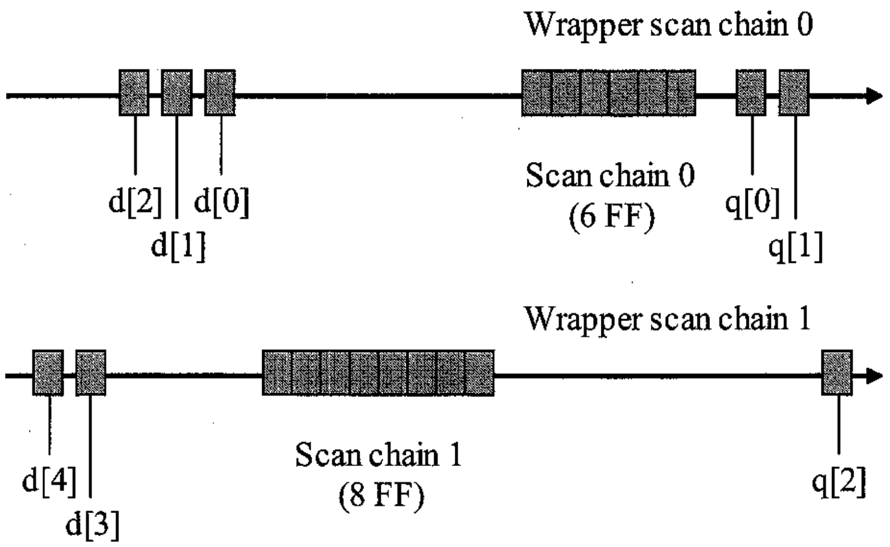


Figure 1.9: The multiple scan chain architecture for Core A in Figure 1.6.

The following description is applicable to the example in Figure 1.7. The first term, $max(...)$, represents the first scan-in cycle and the overlapped scan cycle(s), plus a single capture cycle for each test stimuli. The second term, $min(...)$, represents the final shorter scan-out cycle. Using equation 1.1, the test application time for the example in Figure 1.7 is 57 clock cycles.

For multiple wrapper scan chains, the scan-in depths and scan-out depths for each scan chain can be arbitrary. One possible two-scan chain configuration for the same IP core in Figure 1.6 is shown in Figure 1.9. The scan in-depths of wrapper scan chains (wsc) 0 and 1 are $si_0 = 9$ and $si_1 = 10$, respectively. The scan-out depths are $so_0 = 8$ and $so_1 = 9$, respectively. Because the scan-in operation takes place simultaneously for both wsc 0 and wsc 1, the larger of the scan-in depths, si_1, dominates. Therefore, the overlapped scan-in and scan-out operations take 10 clock cycles. During the final scan-out operation, the number of scan-out clocks is determined by the dominant scan-out depth, $so_0 = 9$.

In general, we can define the maximum scan-in depth for all k wrapper scan chains, $si = max\{si_0, si_1, ..., si_k\}$, and the maximum scan-out depth, $so =$

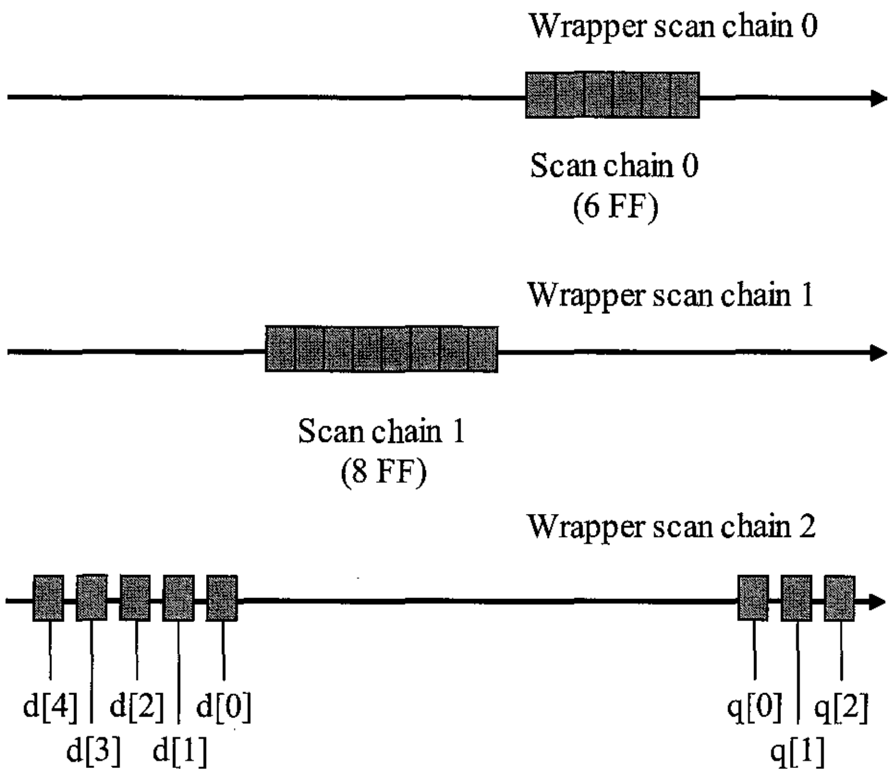


Figure 1.10: Three scan chains architecture for Core A in Figure 1.6. Wrapper scan chain 1 dominates the test application time, therefore larger number of scan chains will not reduce the test time any further.

$max\{so_0, so_1, ..., so_k\}$. The generalized test application time is given by

$$TAT = (max\{si, so\} + 1) \times n_v + min\{si, so\} \tag{1.2}$$

Using equation 1.2, the test application time for the two-scan-chain wrapper configuration in Figure 1.9 with $n_v = 2$ is 31 clock cycles. For three (Figure 1.10) or more wrapper scan chain configurations, the test application time would be 26 clock cycles because the fixed 8-bit internal scan chain dominates.

1.4 Reuse of Functional Interconnects as TAM

The discussions in Section 1.3.1 (Figures 1.5 and 1.6) relate primarily to the concept of a test access mechanism that are added to the chip purely for the use during the test application (i.e. it is only used during the nonfunctional or test mode). There are a lot of research on SoC core-based testing that are relying on the nonfunctional or dedicated TAM. Only a small number of papers considered core-based tests without the dedicated TAM. These related research are highlighted in Chapter 2.

Our work concentrates only on the reuse of functional interconnects as TAM during the test application. The functional interconnects are sometimes called existing interconnects or functional TAM. Unfortunately, these functional interconnects come in various flavors. Bus-based interconnects (hierarchical or non-hierarchical) are the most common. The most recent type of SoC interconnect, the network-on-chip (NoC), has also been considered for high bandwidth requirements. This work considers these three types of functional interconnects as TAM's and develops the necessary support architecture and the corresponding test scheduling algorithms for each case.

1.4.1 Impact on the Design Flow

In Figure 1.1, the design and test flow for the dedicated TAM based test scheme is illustrated. During the process of TAM and wrapper insertion and test scheduling, only the information on the embedded cores is required. For core tests, interconnect information is not required because of the addition of test access mechanisms. The details on the functional interconnect architecture is required only for the interconnect testing.

Figure 1.11 shows the revised design and test flow when reusing the functional interconnect for test data transportation instead of the dedicated TAM. The impact on the design flow is very minimal. The same steps are followed, except during the system-level DfT insertion and test scheduling. The functional interconnect information is required for the wrapper design and test scheduling.

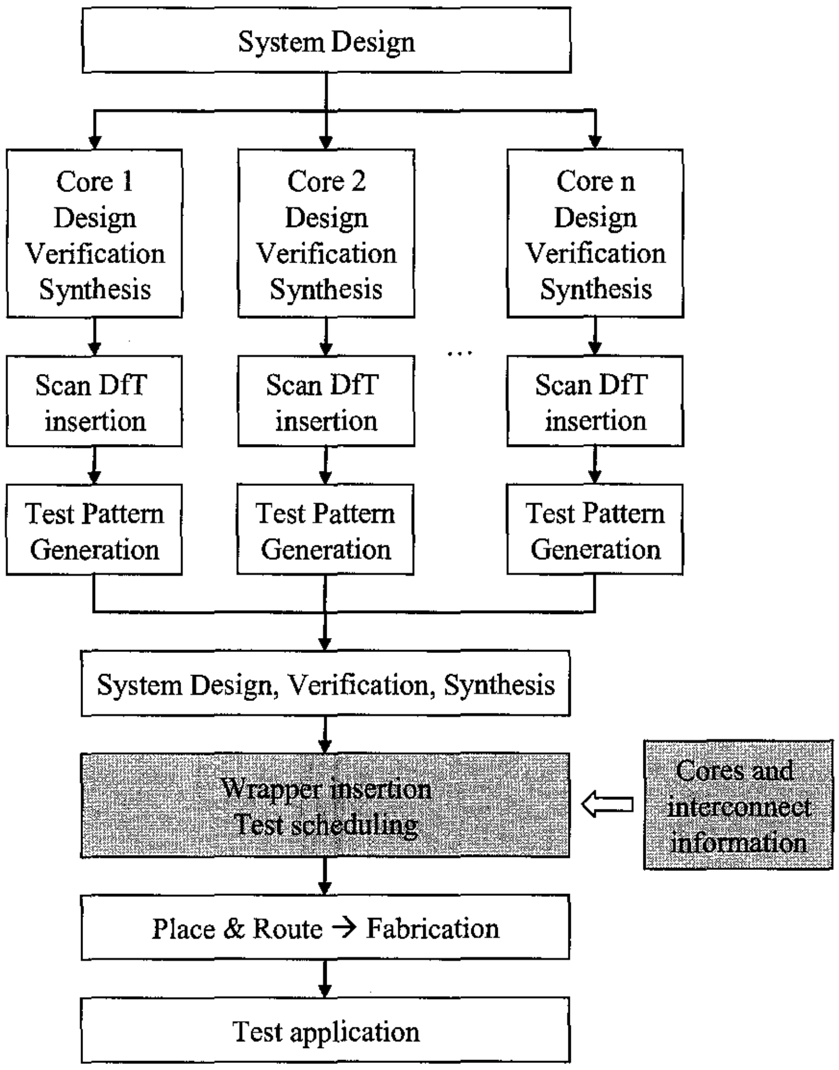

System Design
Core 1 Design Verification Synthesis
Core 2 Design Verification Synthesis
Core n Design Verification Synthesis
...
Scan DfT insertion
Scan DfT insertion
Scan DfT insertion
Test Pattern Generation
Test Pattern Generation
Test Pattern Generation
System Design, Verification, Synthesis
Wrapper insertion Test scheduling
Cores and interconnect information
Place & Route → Fabrication
Test application

Figure 1.11: System design and test flow for interconnect reuse.

1.5 Contributions of This Thesis

This thesis makes three main contributions to the problem of core-based testing of a system-on-chip by reusing the existing SoC interconnect. This thesis focuses on the wrapper design and insertion, and test scheduling, as shown by the grayed blocks in the design and test flow in Figure 1.11. The shared characteristic of all our proposed approaches is that they utilize the existing functional interconnects/interfaces for test purposes. The first step of our research targets the most basic interconnect—the non-hierarchical or flat bus architecture. This is followed by the more advanced hierarchical buses. Finally, we target the new *smart* interconnect called network-on-chip, which is based on the packet-switched Internet protocol network architecture.

Our first contribution is that several variations of core test wrappers for functional interconnect reuse are proposed for specific types of SoC interfaces. We propose an improved version of a core wrapper for bus-based SoC based on the novel buffer-based wrapper architecture by Larsson, *et al.* [6]. Another pair of core test wrappers are designed for a NoC-based SoC, that takes advantage of the NoC communication services, specifically the bandwidth reservation scheme. In addition, the bandwidth matching architecture integrated with the wrapper eliminates the problem present in the wrapper proposed by Amory *et al.* [7].

Second, the thesis proposes the concept of test configuration graphs (TCG), which can be used to systematically and effectively schedule the available resources in a hierarchical bus and multiprocessor system. We proposed five basic types of test configuration graphs, from which any other configuration can be derived. The TCG's are used in the test scheduling algorithm as an effective measure to balance the loads on various bus segments for the test scheduling problem of hierarchical bus based MPSoC.

Third, we are the first to propose a test data transportation scheme that utilizes the NoC effectively, by means of bandwidth sharing. The proposed method drastically reduced the test application time compared to the previously proposed test method for a NoC-based system. Such achievement is made possible by the use of our enhanced NoC-compatible wrappers.

1.6 Thesis Organization

This thesis is organized as follows. The current chapter (Chapter 1) provides a description on the fundamental problems of core-based scan testing of system-on-chips. The problems can be summarized as

- the problem of test access to the deeply embedded cores to provide the test stimuli and to retrieve the test responses, and

- the problem of test time minimization through proper core test scheduling schemes.

Chapter 1 provides a strong motivation for the test data transportation problem in the core-based SoC testing to be mitigated by using the functional interconnects, rather than by using dedicated TAM's. In doing so, the unnecessary area costs and additional routing congestion of the dedicated TAM can be avoided completely.

Chapter 2 begins by considering the test access and test scheduling problems for SoC's with a flat bus (i.e. non-hierarchical) interconnect architecture. The single flat bus poses a problem in minimizing the test application time because of its shared-and-broadcast nature. This shared architecture limits the core test scheduling to a test-one-at-a-time sequential test scheme. To overcome this problem, a buffer-based wrapper architecture is proposed. The addition of buffers allows the shared bus to be used efficiently by means of a packet-based test data transportation, which utilizes the time domain multiplexing (TDM) scheme. The benefit of using the buffer-based test wrapper together with the functional bus on the test application time is evident, as shown by the experimental results presented in this chapter. The proposed test scheduling scheme using the packet-based data transmission is called PAcket Set Scheduling (PASS).

Chapter 3 leads off with a discussion on multiprocessor SoC's (MPSoC) with hierarchical buses. It continues with some explanation on why the PASS algorithm (which was developed specifically for a non-hierarchical bus) does not work efficiently for a hierarchical bus system. This chapter proceeds to address the shortcoming of PASS scheme with an integrated test scheduling methodology called Multi-Processor PASS (MPPASS) for multiprocessor SoC's. As part of the MPPASS scheme, the concept of Test Configuration Graphs (TCG) that represent the MPSoC test architecture is introduced. In short, this chapter expands the methodology proposed in Chapter 2 to be applied to the more generalized MPSoC architecture.

Chapter 4 begins with a brief introduction on the Network-on-Chip (NoC) interconnect architecture. This chapter gives the basic characteristics of NoC topologies, NoC routers, and data switching techniques. Two network-on-chip interconnect architectures—SoCIN and Æthereal—most commonly used in the core wrapper design and core-based test scheduling problems are described.

In Chapter 5, we start by explaining the limitations of the standard IEEE 1500 wrapper, which cannot be used when the NoC interconnect is used as a test access mechanism. The analysis of these 1500 wrapper limitations provides us with the necessary information to construct NoC-compatible wrappers. In order to optimize the bandwidth utilization and minimize the test application time while also minimizing the area cost, two types of NoC-compatible wrappers are constructed. The first wrapper incurs minimal area cost, but wastes the NoC bandwidth under some test configurations. The second wrapper complements the bandwidth-inefficient characteristic of the first wrapper by using more hardware, thus higher area cost.

Having the two wrapper options allows the test designers to balance between area cost minimization and test time minimization in the test scheduling scheme in Chapter 6. The experimental results presented in this chapter evaluates this trade-off. Chapter 6 continues the discussion on NoC-reuse-based test scheduling using a bandwidth sharing scheme—a novel test scheduling technique proposed in this chapter. Compared to other dedicated path-based test scheduling, the dedicated path approach is much more efficient in reducing the test application time, especially with a limited number of I/O ports.

In Chapters 2, 3, 5, and 6, several experimental results are presented and explained to verify the effectiveness of the proposed techniques. Finally, Chapter 7 summarizes the research results presented in this dissertation and highlights some of the important results followed by a brief mention of possible future works.

Chapter 2

Wrapper Design and Test Scheduling for System-on-Chips Using a Functional Bus

2.1 Introduction

System designers are adapting to the system-on-chip (SoC) design methodology because of its efficiency compared to the traditional system-on-board approach. The SoC design technique and its characteristics are briefly described in Chapter 1. The main benefit of the SoC approach is that it can drastically shorten the design cycle by allowing pre-designed cores and their associated test set to be reused. To enable the plug-and-play reuse of tests together with the cores, the IEEE 1500 standard [3, 4] was developed and used by many IP core providers and system designers.

The rapid increase in VLSI design in terms of its functionality as well as the gate/transistor count helps in reducing the design cost. With every new improvement in the manufacturing technology, the cost of fabricating each transistor decreases in inverse exponential fashion (Figure 2.1). The test costs however has been kept almost constant throughout the years. Without a revolution in the way the chips are tested, the costs of a chip will soon be dominated by the test costs.

Furthermore, the use of SoC design methodology introduces several new prob-

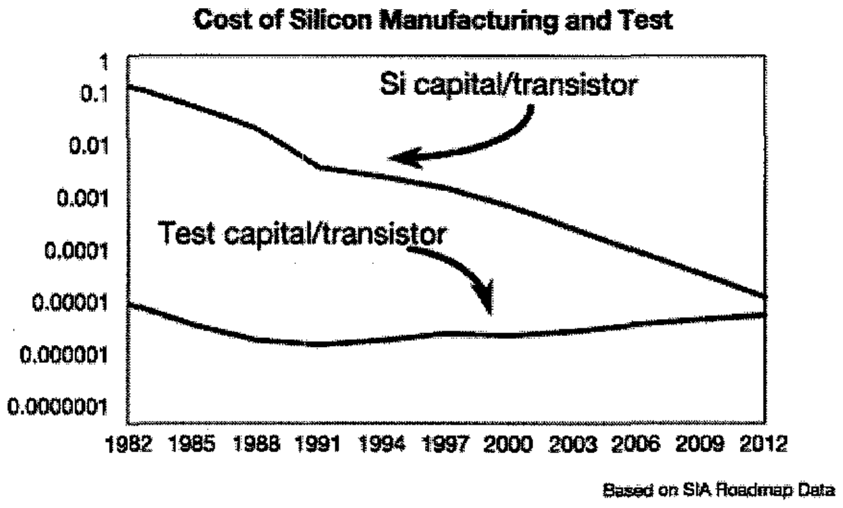


Figure 2.1: Comparison between the cost of silicon and the cost of test of VLSI circuits over a 30-year period.

lems and challenges in testing [8]. First, the cores that are embedded deep inside the silicon chip require a Test Access Mechanisms (TAM) for test data transportation. Several TAM architectures have been proposed such as TestRail [9], Virtual TAM [10], and TAM's based on transparency [11].

Second, the SoC's core-based design requires a mechanism to isolate the cores during test. This is achieved by the use of core wrappers [3, 9]. Third, the cores can either be tested sequentially at the cost of longer test application time, or in parallel at the cost of larger area overhead and power dissipation. In this regard, various test scheduling solutions have been proposed [6, 12–50]. The core test scheduling approaches proposed by [13–21, 27] rely on a dedicated TAM. This extraneous TAM is consistently added to the SoC for the sole purpose of delivering the test vectors from external automatic test equipment (ATE) to the core under test (CUT). In general, the test application times resulting from these approaches are bounded by the lower bound derived in [21].

Among these approaches, various optimization methods stand out such as using *k-tuples* to represent a test schedule [29], 2D-bin packing [17, 19, 51], 3D-bin packing [13], several types of linear programming [30–32], simulated annealing

[33], preemptive scheduling [14, 34], and a schedule based on the TAM-routing cost [35], a DFT selection algorithm [36], a wrapper/TAM co-optimization [12], and a multi-frequency TAM [37]. In order to further minimize the test application time, reconfigurable wrapper [14], virtual/reconfigurable TAM [16, 38, 39], test vector sharing and broadcasting [27, 52], and defect-probability driven [40, 41] approaches have also been proposed.

Taking into account the special characteristics and limitations of the SoC chip, power-constrained scheduling [17, 18, 34, 42–46], layout-constrained scheduling [47] and test scheduling for multi-clock domain SoC's [15] and hierarchical SoC's [19, 48–50] are also important. Equally important is the work on test planning and design space exploration [20, 53] which tackle the test problem at an earlier stage. All these test scheduling and optimization works rely on the introduction of a dedicated and extraneous TAM for the test data transportation.

Regardless of how efficient the test schedule optimization, wrapper optimization, or TAM optimization algorithms are, the idea of adding a dedicated TAM for test data transportation by itself requires considerable area overhead. In addition, the long TAM wires increase the routing congestion. With small feature sizes below 90 nm, the long wires are highly potential spots of production defects. Defects in TAM's would prevent any cores from being properly tested, thereby affecting yield. To avoid relying on TAM's, the existing functional communication architecture should be used as an alternative to the extraneous TAM for testing purposes. The test scheduling methods proposed in [13–21, 27] cannot be applied to this new problem, which has a single shared bus where each bus wire cannot be individually assigned according to the optimum test schedule. Cores can only be tested sequentially by using all functional bus bit width.

In order to maximally reuse the existing chip resources for testing, the authors in [22] described a method based on consecutive transparency, where test access paths are formed by creating transparent paths through existing functional connections between the SoC cores, thereby reusing most of the existing interconnects. However, the drawback of the proposed approach is that it is intrusive; sometimes, establishing the paths requires modification to the core internals, which might affect the critical paths of the cores.

Several test strategies which utilize the functional bus [6, 23–26, 28] have been proposed; they are further discussed in Section 2.2. However, these methods do not consider the test scheduling at the packet level, which is necessary when reusing the functional buses for test data transportation. In this chapter, we introduce our proposed approach [54, 55], which explicitly schedules the transportation of data packets through the functional bus that carry the test vectors and responses. The test application times are obtained through complete packet-level simulations of the test data transportation. The proposed approach is unique because of the explicit packet transportation schedule, in addition to its ability to make use of the functional bus effectively.

We begin with a review of some works related to the bus reuse strategy in Section 2.2. In Section 2.3, a motivational example is given, followed by a brief technical overview in Section 2.4. In Section 2.5, the support architecture design for the efficient utilization of the functional bus during testing is described. Section 2.6 elaborates the methodology to develop an efficient test schedule using the functional bus. In Section 2.8, we thoroughly evaluate our methodology experimentally. Finally, a brief set of conclusions is offered in Section 2.9.

2.2 Related Work

This section describes some works on the utilization of the existing functional interconnect rather than an added interconnect as TAM. To differentiate the two types of test data transportation approaches, we define the following terminologies:

Definition 2.1 *Dedicated TAM* is a set of dedicated wires that are added to the SoC for the test data transportation between an ATE and all the SoC cores during nonfunctional or test mode.

Definition 2.2 *Functional TAM* refers to the existing SoC's functional interconnects which are transformed and reused for the test data transportation during the nonfunctional or test mode.

Among the earliest literature on the use of functional TAM's for SoC testing stands out the paper by Papachristou *et al.* [24], in which, an embedded microprocessor is used as the test controller. The test data from an external tester are loaded into the embedded memories through a direct memory access controller (DMA) and delivered to the core under test by the embedded processor through the functional interconnect. The test responses are evaluated by embedded signature analyzers in order to minimize the load on the processor and the interconnect.

A more comprehensive methodology on the functional TAM approach was discussed by Harrod [23]. The paper discussed the test application strategy for various types of test requirements for the embedded Intellectual Property (IP) cores. Test access to the Core-Under-Test (CUT) is provided by the Test Interface Controller (TIC), which is part of the AMBA specifications. An external ATE is used to deliver the test vectors through the TIC interface. At each core, a test wrapper is required to isolate the core from its surrounding.

Krstic *et al.* [25] presented a similar methodology, where the embedded processor first tests itself by executing a set of instructions using a software-based self test (SBST) methodology [56]. Subsequently, the processor tests the bus and the other IP cores. Huang *et al.* [26] presented a similar core-based test approach for PCI bus based SoC's. The test application is performed by the test support architecture at every CUT. Software-based weighted random patterns [57] are generated for each CUT. These approaches [23–26], however, do not propose a test scheduling technique for a complete SoC to show their effectiveness. To enhance the software-based test generation, a specialized set of test instructions is also introduced [58].

Larsson *et al.* [6] proposed a buffer-based test support architecture to enable parallel testing of core-based SoC's. As opposed to the embedded processor-based approach by [24–26], the test control is performed by an embedded finite state machine based controller which costs additional hardware, proportional to the volume of the test data.

All the proposed methods [6, 23–26] target the structural test of the logic cores. The tester (an embedded processor or an external tester) utilizes the functional buses to transfer the structural test vectors to the respective CUT's. The authors in [58] targets the functional tests of embedded processor and logic cores by implementing an instruction-level design-for-testability (DfT) and specialized test instructions to aid in increasing the fault detection and shortening the test application time.

The use of packet-switching architecture for core-based testing has been proposed by Aktouf [59] for a multiprocessor system with a homogeneous network-based architecture. Vermeulen *et al.* [60] shows that an NoC can be effectively reused for core testing as well as system verification. Several NoC-reuse based test strategy have been published in the recent years, among them are from Cota *et al.* [61], Amory *et al.* [62], and Liu *et al.* [63]. All these approaches, however, do not propose an explicit transportation schedule for each test packet; instead, dedicated access paths are allocated assuming that the test data and response packets can be streamed between the tester to the CUT's. Applying these approaches on a shared functional bus results in a sequential testing of the SoC cores.

Nahvi *et al.* [64] proposed a Test Access Mechanism (TAM) architecture based on a packet switching communication network, called NIMA (Novel Indirect and Modular Architecture). NIMA provides access to the cores for test data delivery by means of multi-level routers and communication channels similar to bus-based TAM's. Packet delivery is achieved by breaking the test data into smaller units called packets, each of which containing the necessary control information required to successfully deliver the data through the routers. Typical control information includes the synchronization bits, the destination address, and the length of the address field. Initial analysis of the packet routing architecture shows an improvement in terms of core access time and the total wire length as compared to the bus-based TAM's.

A hybrid TAM architecture which uses the existing functional bus in addition to extra TAM's was proposed in [27]. In this approach, the functional bus is converted into a bundle of TAM's, by adding some logic to make the bus wires controllable and observable to external testers. As a result, each functional bus wire can be individually controlled and assigned to cores for the test data transportation, similar to the added TAM. To further minimize the test application time by optimizing TAM utilization, shared test vectors are broadcast to multiple cores.

In [26], the authors propose a test interface architecture between PCI buses and CUT's. The CUT's are tested using pseudo-random test vectors, generated by the embedded processor. In [6], a buffer interface between a functional bus and a CUT is proposed, while the control of test application is performed by a Finite State Machine (FSM) based controller. The hardwired controller has in [6] has two main weaknesses compared to our approach: (1) the area cost is proportional to the the volume of test data, and (2) the FSM-based test schedule is fixed, making it impossible to change. This flexibility is especially important during the hardware debugging stage. The test responses, on the other hand, are not transported through the bus to the test sink. Instead, they are compressed by local embedded multiple input signature registers (MISR), which could cause aliasing. Furthermore, the MISR's incur hardware overhead.

Due to the use of MISR's, the test application time of [6] should, in most cases, be shorter than our approach, which transports the test responses back to the tester. However, the use of software-based test program in our proposed approach has the advantage of being flexible, and does not incur hardware overhead other than the buffers. The role of the FSM-based test controller in [6] can be replaced by an external tester or an embedded processor. This is further discussed in Section 2.4

In this chapter, we illustrate our power-constrained SoC testing approach which utilizes the functional TAM for the test data transportation. In order to take advantage of the functional TAM, we approach the problem from two angles, namely, a support architecture design framework and an algorithmic framework. In the process, we show how our approach greatly simplifies the test program, one of the primary strengths and differentiators of our proposed methodology. Such a simplification is attained through the support of an efficient test architecture, which includes appropriate timing control circuitry.

2.3 Motivation

Let us look at some of the possible scenarios regarding packet based test delivery utilizing the shared functional TAM. To ease description, let us denote each of the small test data units as a *test packet*. Figure 2.2 illustrate a sequence of

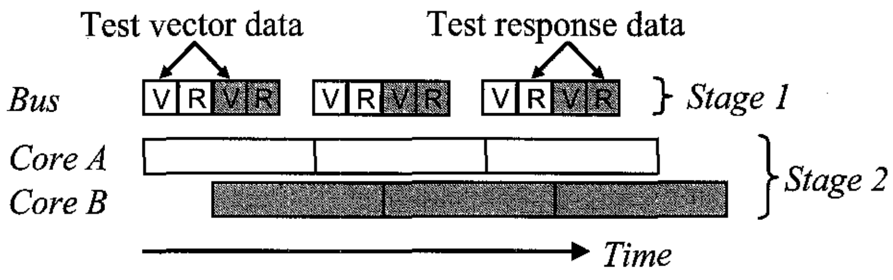


Figure 2.2: Test data transportation using packet-based delivery on the functional TAM. An *R* packet carrying the test response data is returned to tester after every *V* packet, which carries test vector data.

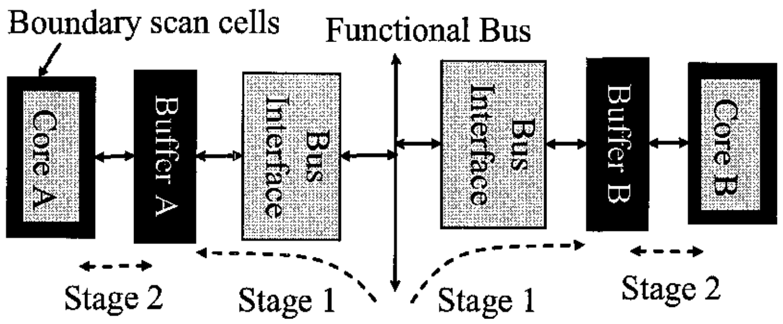
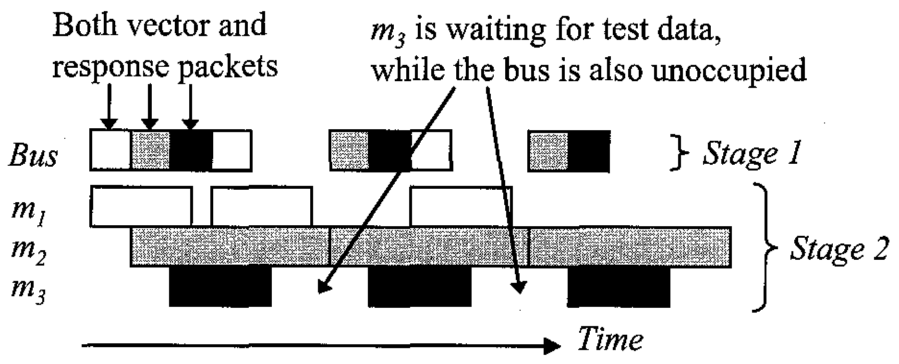


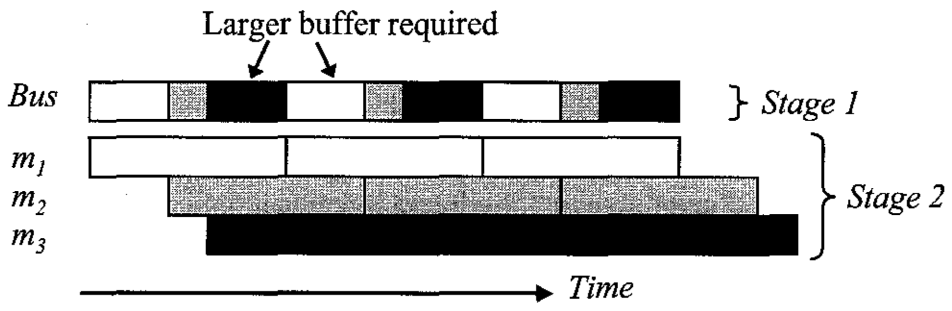
Figure 2.3: Buffer-based test architecture enables parallel test application while utilizing a shared functional TAM.

events when test packet are transported between a tester and two CUT's, *Core A* and *Core B*, which are interfaced to the functional TAM through dedicated local buffers *Buffer A* and *Buffer B*, respectively, in Figure 2.3. *Bus* and *Core* A/B represent the activities on the functional TAM and at the CUT, respectively. Once a packet carrying test vector data, v_t, (labeled *V*) is received by the local buffer, the test response packet (labeled *R*) from a previous test vector, v_{t-1} is returned in the next time slot.

The *Round-robin* packet delivery schedule in Figure 2.2 is a reasonable first attempt at scheduling the test delivery because of its fair allocation of the bus.


Both vector and response packets
m_3 is waiting for test data, while the bus is also unoccupied
Bus
m_1
m_2
m_3
Stage 1
Stage 2
Time

(a) Fixed packet size



(b) Variable packet sizes

Figure 2.4: Effect of repetitive packet delivery sequence for different packet sizes.

Figure 2.4(a) shows a similar delivery pattern for three unidentical CUT's C_1, C_2, and C_3. For each CUT c_i, each packet will go through two separate stages of transfer (illustrated by Figure 2.3). First, it is delivered from a tester to the buffer through the functional TAM (labeled *Bus* in Figure 2.4, where each time slot represents both V and R packets). On the second stage, the packet is transferred from the buffer into the scan chains (labeled c_i in Figure 2.4 to illustrate the concurrent activities of all CUT's c_i in stage 2). A test response packet is returned to the tester after every successful reception of a test vector packet by the buffer.

Stage 1 of the subsequent test packet for a CUT can only begin, to avoid buffer overflow, after stage 2 of the previous test packet for that CUT has been completed. Furthermore, since stage 1 uses a common bus, only one test packet can be in stage 1 at any given time. Stage 1 and stage 2 are also referred to as *test delivery* and *test application*, respectively, for the test packet.

Figure 2.4(a) shows CUT C_3 idle, waiting for test data because the test packet for C_3 cannot be delivered until the test packet for C_2 has been delivered. However, the test packet for C_2 cannot be delivered until the test application of the previous packet of C_2 has been completed. Consequently, C_3 is starved for test data and at the same time the bus remains idle while waiting for C_2 to complete test application even though C_3 needs test data. An analogous situation holds for C_1. CUT C_2, on the other hand, always receives its test data in a timely manner at the expense of starving C_1 and C_3.

The problem can be remedied by increasing the packet size for C_1 and C_3, as in Figure 2.4(b). However, this quick fix implies that larger buffer spaces are required for C_1 and C_3 to store the larger packet sizes. We can reduce packet sizes for all cores, but the minimum packet size for each core is constrained by the core with the smallest packet size (i.e. C_2). Further reduction in packet sizes for C_1 and C_3 would reintroduce the problem illustrated in Figure 2.4(a). The two scenarios illustrated by Figure 2.4 above are not the only optimization problems that have to be solved to make core-based testing using the functional TAM attractive. An additional challenge stems from the fact that packet sizes cannot be arbitrary, because data delivery is conducted through a discrete number of bus wires.

Figure 2.3 shows the buffer interface between the bus and the core, which is considered in this chapter. There are several different variables that can affect test scheduling—bus frequency, bus width, scan frequencies, number of scan chains, and volume of test data for each core. All these variables contribute to the efficiency of the test schedule.

2.4 Technical Overview

Even though a functional TAM and a dedicated TAM may be similar in many ways, the underlying issues that need to be considered are completely different. We can broadly categorize them into two:

- Support architecture for test data delivery
- Algorithmic framework for efficient test scheduling

A functional TAM differs from a dedicated TAM because every functional TAM wire is connected to every embedded core; it cannot be committed to multiple cores simultaneously like a dedicated TAM by assigning subsets of TAM wires to different cores. Without a buffer-based test support mechanism (Figure 2.3) similar to the test buffer in [6, 26], only one core can be tested at a time. Furthermore, in the conventional approach of dedicated TAM-based SoC testing, test control timing, including scan and capture clock generation, is provided by the external ATE. Therefore, the synchronization of test vector availability at the scan input and scan/capture clock generation is trivial since the ATE retains full control of all the event sequences for every core under test. When utilizing the functional TAM, this control timing needs to be performed on-chip, posing a research challenge, which we subsequently address in this chapter.

The test source/sink to a functional TAM can either be an external ATE or an internal programmable block. Unlike dedicated TAM's, the ATE cannot be connected directly to the functional TAM wires, but through a test interface port shown in Figure 2.5. For example, ARM's AMBA [65] bus architecture provides a Test Interface Controller (TIC), which communicates to the ATE and the CUT using the functional read/write transactions. The TIC acts as an intermediary by relaying the the vector data packets from the ATE to the CUT and vice versa for the response packets.

In order to utilize the programmable core as a test source/sink, several issues need to be resolved, which among others include testing of the programmable core itself, and loading of the test stimuli to and and offloading of the test responses from the programmable core. In this chapter, we will utilize an external ATE (connected through a TIC) as a test source and sink, therefore. these issues regarding the programmable core are not addressed here.

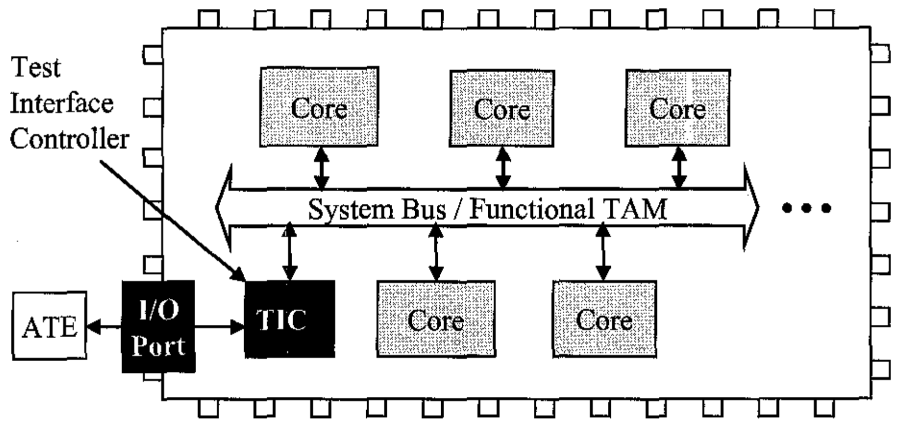


Figure 2.5: Interfacing an ATE through a Test Interface Controller (TIC) port for a functional mode test data transfer through the functional TAM.

Test scheduling that enables the reuse of the functional bus as, functional TAM's involves three steps. First, breaking the test set into subsets capable of efficiently utilizing the bus. Second, scheduling all tests for every core with the objective of test time minimization, and third, generation of a test program which will execute in real-time by the tester to perform test application for all the SoC cores. In order for the benefits of utilizing the functional TAM for testing to outweigh its counterparts in the dedicated TAM approach, the test architecture needs to support the algorithmic framework, and vice versa. Otherwise, problems such as bus underutilization arise because of the required arbitration between different cores, and improper test schedules causing certain cores to starve of test data while other cores may be hogging the bus, resulting in prolonged test application time.

Test data overflow/underflow is a particularly serious potential problem, unless the necessary timing synchronization or interlock between the tester and the cores under test is provided. The test support architecture should be designed to also provide timing synchronization, while minimizing the hardware overhead.

2.5 Test Support Architecture

Buffer based test architecture was proposed by [6] in order to enable concurrency of core-based testing using a shared functional TAM. Our general test architecture with buffers similar to [6] is shown in Figure 2.3 with the corresponding test delivery and test application timing diagram similar to Figure 2.4.

Figure 2.6 shows the detailed architecture of the interface between the functional TAM and the core through the functional bus protocol interface. Both the functional connections (solid lines) and design-for-testability (DFT) connections (dotted lines) are shown. The components shown in solid black shading are the proposed buffer-based DFT architecture. Boundary cells (BC) are added to the core PI/PO's in order to isolate the core during test. The wrapper scan chains are formed by chaining the input BC's, internal scan chains (ISC), and output BC's, in that precise order. Bidirectional I/O's are treated similarly to the ISC's since the BC's scan inputs, and not functional inputs, are used. Bidirectional I/O's are not shown in Figure 2.6 to avoid clutter.

The next section (2.5.1) explains one of the most popular methods of forming the wrapper scan chains that minimize the test application time under given constraints. The concept of scan chain design is important in order to completely understand and appreciate the core wrapper design.

2.5.1 Wrapper Scan Chain Design

The authors in [66] provided a detailed analysis on test time for three types of scan chain architecture, namely *multiplexing, daisychain* and *distribution* architectures. The fundamental concept of test time minimization through scan chains design can be summarized here.

A scan test consists of three phases: (1) scanning in of the stimuli, (2) normal execution, or commonly known as the capture cycle, and (3) scanning out of the captured responses. As briefly described in Section 1.3.1 for the standard wrapper in Figure 1.6, the components that make up a scan chain are

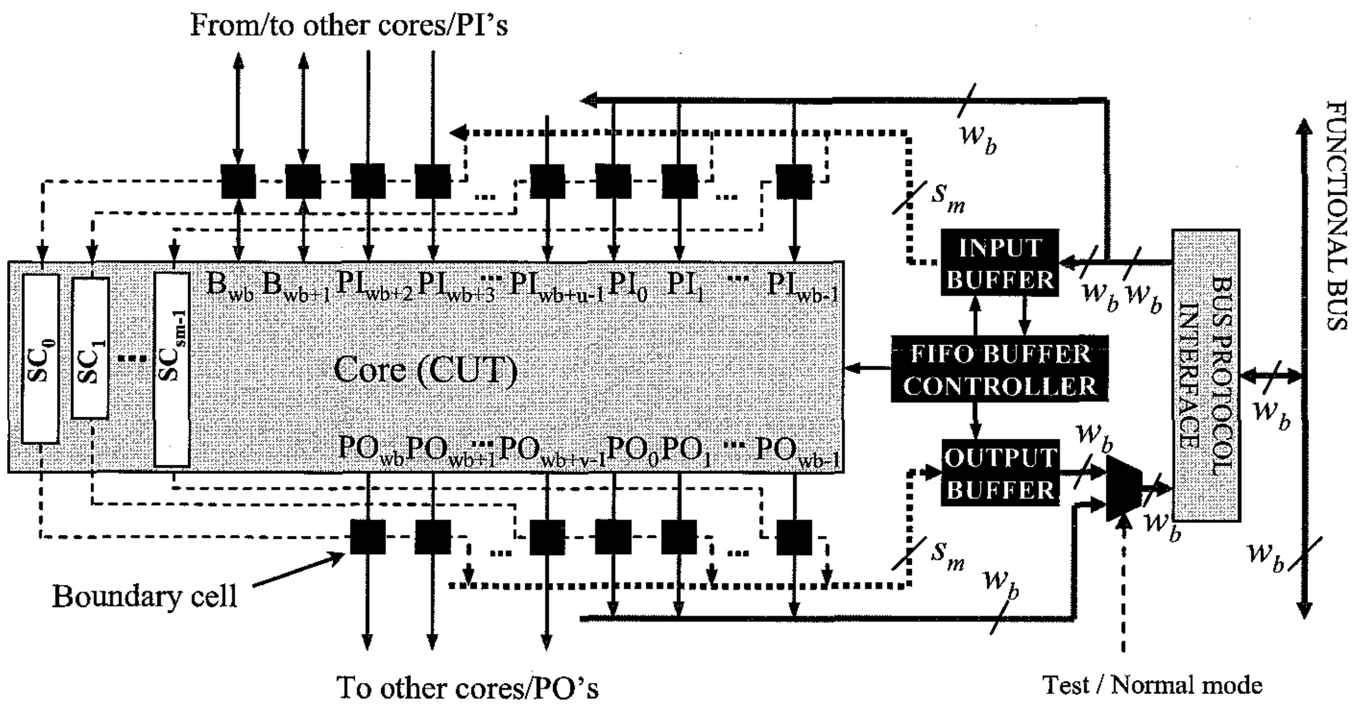


Figure 2.6: Core test architecture with input and output buffers to temporarily hold the test vectors and test responses, respectively.

- input wrapper boundary cells, whose quantity is the same as the number of the core's primary inputs, n_{ip}
- output wrapper boundary cells, whose quantity is the same as the number of the core's primary outputs, n_{op}
- bidirectional wrapper cells, whose quantity is the same as the number of the core's primary bidirectional input/outputs, n_{bi}
- k internal scan chains, whose total length is $\sum_{x\in[1..k]} l_x$, where l_x is the length of each individual internal scan chain x.

The test stimuli must be available at the core's primary inputs (inputs and bidirectionals) as well as the internal registers, which are now augmented into scan cells, before the capture cycle. Therefore, the scan-in operation would take si_1 scan cycles (equation (2.1)), assuming a single scan chain architecture. Similarly, the scan-out depth would be so_1 cycles, given in equation (2.2). The scan-in and scan-out items are illustrated in Figure 2.7.

$$si_1 = n_{ip} + n_{bi} + \sum_{x\in[1..k]} l_x \tag{2.1}$$

$$so_1 = n_{op} + n_{bi} + \sum_{x\in[1..k]} l_x \tag{2.2}$$

Further, after the first capture cycle, the scan-in of new test stimuli can be done concurrently as the scan-out of the responses associated with the previous stimuli. Therefore, an equal number of scan-in and scan-out cycles is the most desirable. For a single wrapper scan chain, the solution is trivial. With multiple scan chains, the problem of scan chain design becomes a complex scan chain distribution problem such that the test application time for the core is minimized. The test application time, T, is given by equation (2.3), si and so are the maximum scan-in and scan-out depths, respectively, and n_v is the number of test stimuli or test vectors.

$$T = (max\{si, so\} + 1) \times n_v + min\{si, so\} \tag{2.3}$$

2.5.2 Operation of the Buffer-based Test Architecture

During the test application, the test data are delivered to the input buffer and then scanned into the scan chains. At the same time, the test responses are

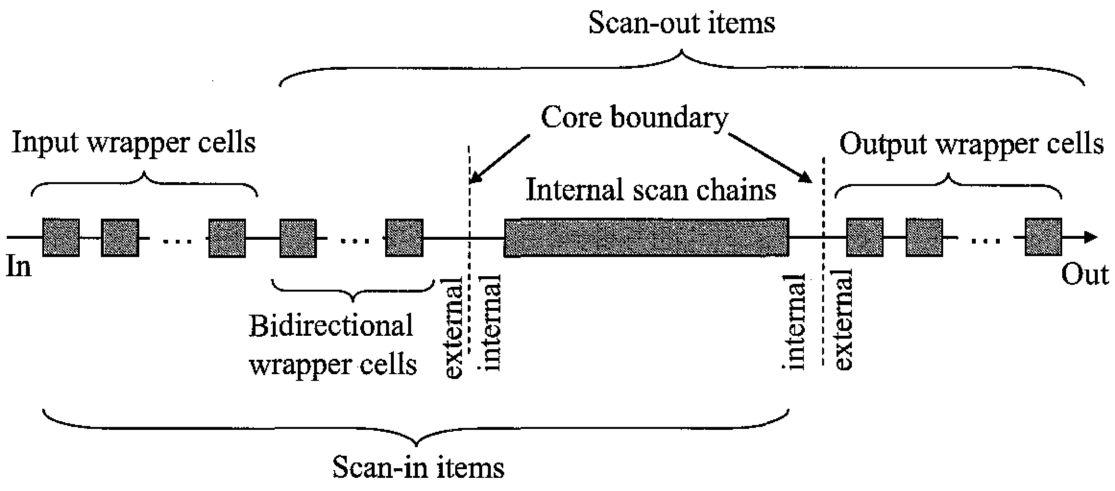


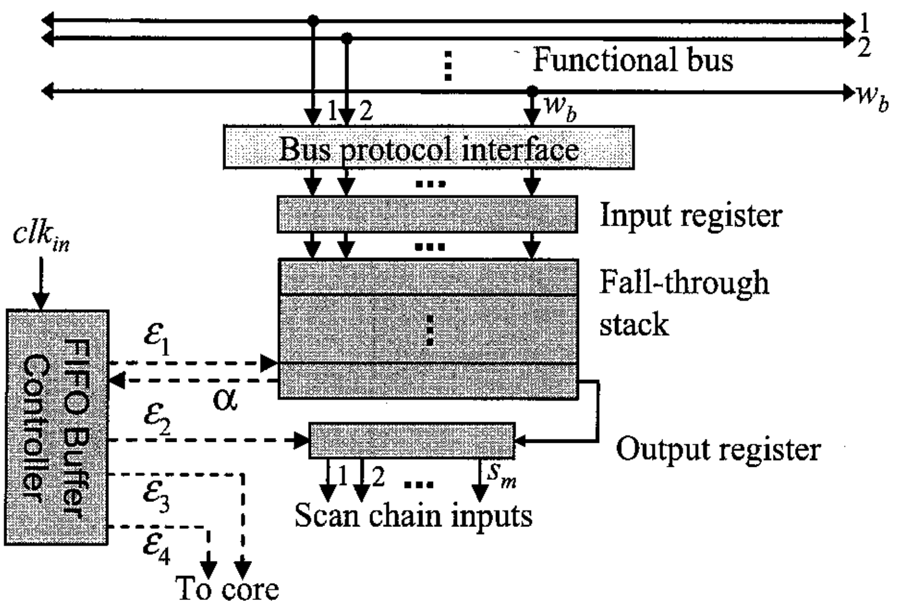
Figure 2.7: Break down of the scan chain architecture.

scanned out and stored in the output buffer before being retrieved by the tester for analysis.

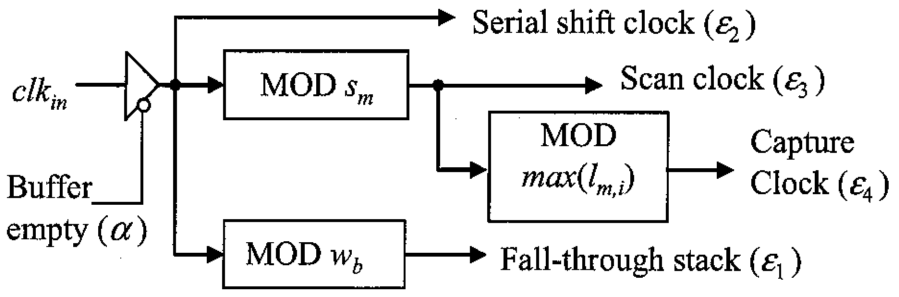
The buffer consists of four main components—input register, output register, fall-through stack, and FIFO buffer controller—as shown in Figure 2.8(a), illustrating the input buffer and the corresponding first-in first-out (FIFO) buffer controller. The output buffer (not illustrated) has identical structure as the input buffer but with reverse data flow. The input register latches data from the bus. Upon registering a full status bit for the input register, the top of the stack copies the data from the input register if its status bit indicates that it is empty. After copying, the input register status bit is cleared, preparing it for the next cycle of data from the bus. The stack will subsequently go through the fall-through stages which will bring the data to the lowest empty slot.

The output register is composed of s_c bits, where s_c is the number of wrapper scan chains for core m, possibly differing from the bus width, w_b. It is interfaced directly to the scan chain inputs. The output register is designed to support this mismatch in bus width and wrapper scan chains. Therefore, it can be easily adapted to any number of scan chains regardless of the bus width.

The test data is serially shifted out from the bottom of the stack, and shifted into the output register. The FIFO buffer controller keeps track of the number



(a) The proposed buffer architecture interfaces the functional TAM and the core scan chains.


Serial shift clock (ε_2)
clk_{in}
MOD s_m
Scan clock (ε_3)
MOD $max(l_{m,i})$
Capture Clock (ε_4)
Buffer empty (α)
MOD w_b
Fall-through stack (ε_1)

(b) FIFO buffer controller generates test control signals to the buffer and the CUT when test data are received.

Figure 2.8: Test buffer architecture.

of bits being serially shifted into the output buffer, n_{si}, and the number of bits being serially shifted out of the bottom stack, n_{so}. Serial shifting is clocked by the tri-stated clock signal (labeled ϵ_2 in Figure 2.8(a) and Figure 2.8(b)). When

n_{si} equals s_c, the FIFO controller generates a scan clock ϵ_3 to scan in the contents of the output buffer into the scan chain, whereupon new data is shifted into the output buffer. When n_{so} equals w_b, the FIFO controller generates a signal ϵ_1 to fill in the bottom of the stack with new data.

The FIFO controller also keeps track of the number of scan clocks already generated. When this number is equal to the longest scan chain in the core, $max(l_{m,i})$ for all i scan chains in core m, a capture clock ϵ_4 is generated. The FIFO controller can be implemented using three modulo counters, i.e., MOD s_c, MOD w_b and MOD $max(l_{m,i})$ as illustrated in Figure 2.8(b). The required input for this circuit is clock clk_{in}, whose frequency value is the product of the number of wrapper scan chains, s_c, and the scan frequency, f_c. The same FIFO controller is used for both input and output buffers because of their inverse operation. This eases timing synchronization between the input and output buffers.

The proposed buffer architecture offers two distinct advantages. First, the test application at the core operates asynchronously with respect to the availability of test data in the buffer. When empty, the buffer disables the control signal generation by means of status signal α, which is an input to the FIFO controller. Because of the asynchronous scan and capture clock generation by the FIFO controller, the buffer can accommodate unpredictable delivery time of the test vectors, thus handling the synchronization issue. As a result, the scan clock and the bus clock can be decoupled. Such decoupling enables the proposed test mechanism to utilize a bus frequency higher than the scan frequency. Such a capability is lacking in a dedicated TAM-based approach because TAM wires are connected directly to the scan chains.

The second advantage is that the buffer allows the test data to be delivered in chunks of any arbitrary multiple of bus width. This flexibility proves to be quite useful in optimizing the test schedule, in addition to minimizing the buffer area overhead.

On the other hand, the multiplexer at the output buffer (Figure 2.6) introduces one additional gate-delay, compared to the standard IEEE 1500 architecture. However, the impact on functional timing between the functional bus and the core is minimized because the buffers are added in parallel to the functional paths. Another drawback is that gated scan clock is used to disable the scan operation when the buffer is empty, which might affect the delay characteristics of the clock tree.

Due to the DFT architecture, which interfaces the core directly to the functional bus, the proposed methodology is not applicable to embedded cores that are not directly accessible from the bus. One possible way of mitigating this is by introducing bypass interconnects between the core and the bus; this issue is out of the scope of this chapter therefore not discussed.

In Section 2.6, the scheduling of test vectors and responses transportation for the embedded cores is discussed. It is assumed that the FIFO buffers and controllers are fault-free, therefore not the target of testing. The test of these DFT architectures can be either done in an integrated fashion, or independent of the core tests (i.e. in priory). In this chapter, the latter is assumed.

2.6 Packet Delivery Scheduling Algorithm

In this section, the issues related to packet delivery scheduling discussed in Section 2.3 are addressed. The packet delivery schedule that minimizes the test application time is developed with two objectives:

1. Minimization of the total required buffer size.
2. Maximization of bus utilization.

The above objectives are sought while at the same time ensuring that all cores receive the test data in a timely manner. In order to satisfy these twin objectives, the buffer size for each core and the test delivery sequence need to be optimal.

The scheduling algorithm consists of two hierarchical steps. The first step (described in Section 2.6.3 and 2.6.4) is the grouping of cores which can be tested simultaneously under a maximum power constraint. In the second step (defined in Section 2.6.5 and 2.6.6), for each group of cores, the optimum number of packets (and the corresponding packet size) for every core is determined. Each of these packets is then scheduled for delivery through the functional TAM.

In this section, the algorithmic framework is discussed in terms of the two hierarchical steps above. We start by defining a set of nomenclature useful in describing the methodology.

2.6.1 Terminology

Definition 2.3 A *test packet* is composed of a number of bits of test data delivered to a core by the tester, in one burst transfer through the bus.

Definition 2.4 Due to the delivery of test packets through the w_b-bit wide functional TAM, the number of bits of test data that makes up a test packet is typically $p_c \times w_b$, where p_c is denoted as the *packet size.*

Definition 2.5 A *test group* consists of a subset of cores in an SoC that are tested simultaneously.

Definition 2.6 A *packet set* is composed of a series of packets delivered to all cores $c_i \in M_G$, where M_G is a test group. Several identical packet sets can be cascaded to form a *packet schedule* consisting of all packets for all cores c_i to complete the test of M_G. Figure 2.2 shows a packet set $\{Core\ A,\ Core\ B\}$ repeated three times to form part of a test schedule.

Definition 2.7 A core c_i is said to have a *split ratio* of k, if k packets are scheduled for core c_i in one packet set. In other words, it means that core c_i will have k times the number of packets of the smallest cores with a split ratio of one. The core is also called a *split-k core.*

Definition 2.8 The *scan rate* (R_c) is the speed at which the test vectors are loaded into the wrapper scan chains and the test responses are shifted out of the wrapper scan chains in bits per second (bps). A core with s_c wrapper scan chains and f_c scan frequency has a scan rate of $R_c = s_c \times f_c$.

2.6.2 Power and Heat Dissipation Problem

Designers can easily increase the computation power of today's SoC to meet the computational needs of the system. The major problem remains to be the excessive heat that these individual power-hungry cores generate. A design with several of these cores could easily damage the chip through overheating, if clever and necessary power management is not utilized.

The cost of test is linearly proportional to the amount of time taken to test the chip. Therefore, it is imperative that the test application time be as short as possible. This is almost always the main objective of any test scheduling strategy. Since shortening the test time requires the concurrent tests of multiple power-hungry cores, some sort of strategy is required to optimize the test time without violating any of the design constraints such as the upper bound on power and temperature. This trade-off demands the application of smart and efficient test strategies.

Each SoC core dissipates a certain amount of heat. In typical SoC testing, due to the design characteristics such as heat dissipation and current carrying capacity of wires, a limit is imposed on power dissipation that a circuit can tolerate without causing permanent damage to the chip.

In this dissertation, instead of considering the actual amount of heat dissipated by each core, we limit the power consumption of the whole chip during test. The total power dissipation at any given time t is the sum of power dissipations of all cores that are tested (or active) at time t. To optimize the test scheduling scheme, we utilize a scheme that optimizes the scan frequency for each of the circuit under test.

2.6.3 Scan Frequency Reductions

In the illustrative example in Figure 2.9, the maximum power dissipation is given by Q_{max}. Given that the sum of power dissipations for all three cores C_1, C_2, and C_3 exceeds Q_{max}, the three cores cannot be tested simultaneously. This is illustrated in Figure 2.9(a) where core C_3 is tested after the tests of cores C_1 and C_2 without exceeding the maximum power dissipation, Q_{max}. Each rectangle labeled C_i represents the power dissipation (vertical dimension) and test duration (horizontal dimension) required to test core C_i.

The original test configurations do not allow simultaneous tests of all cores C_1, C_2, and C_3 without violating the power constraint. However, as shown in Figure 2.9(b), if the power-time rectangles for C_2 and C_3 can be reshaped while keeping the area inside the rectangles constant, all cores can be tested concurrently, resulting in shorter total test application time, without violating the Q_{max} constraint. This power-time rectangle's shape transformation by means of changing the scan frequency has been used and discussed by [15] in their dedicated TAM-based SoC scheduling methodology.

In order to do this shape transformation, we first look at the relationship between scan power and scan test time, which defines the dimensions of the power-time rectangles. A dynamic power dissipation—the dominant component of total power dissipation—due to charging and discharging of output capacitance of every gate [67] in the chip-under-test is given in equation (2.4).

$$Q_{dyn} = 0.5 \times C_{load} \times V_{DD}^2 \times f_{sys} \times N_G \tag{2.4}$$

where C_{load} is the load capacitance, V_{DD} is the supply voltage, f_{sys} is the system clock frequency, and N_G is the total number of gate output transitions. During a scan test with a scan frequency of f_c, the dynamic scan power (Q_c) and test application time (T_c) for a core c are:

$$Q_c = 0.5 \times C_{load} \times V_{DD}^2 \times f_c \times N_G \tag{2.5}$$

$$T_c = (max_k\{l_{c,k}\} + 1) \times \frac{n_{vc}}{f_c} \tag{2.6}$$

where $l_{c,k}$ is the individual length of k internal scan chains, and n_{vc} is the number of test vectors for core c. If the scan frequency is reduced to $f_c^* = f_c/2$, the new scan power (Q_c^*) and test time (T_c^*) becomes:

$$Q_c^* = 0.5 \times C_{load} \times V_{DD}^2 \times \frac{f_c}{2} \times N_G \tag{2.7}$$

$$T_c^* = (max_k\{l_{c,k}\} + 1) \times \frac{n_{vc}}{f_c/2} \tag{2.8}$$

The effects of reducing the scan frequencies are that the TAT increases and the test power decreases, correspondingly while keeping $Q_c \times T_c = Q_c^* \times T_c^*$. Variable N_G is constant because under different scan frequency, the same test set is still applied. This reduction in test power (at the cost of prolonging the test time) is important when trying to schedule concurrently as many cores as possible under a given power constraint, Q_{max}.

Changing the test frequency requires either a phase-locked loop (PLL) or a frequency divider. Nevertheless, the implementation of the frequency divider is

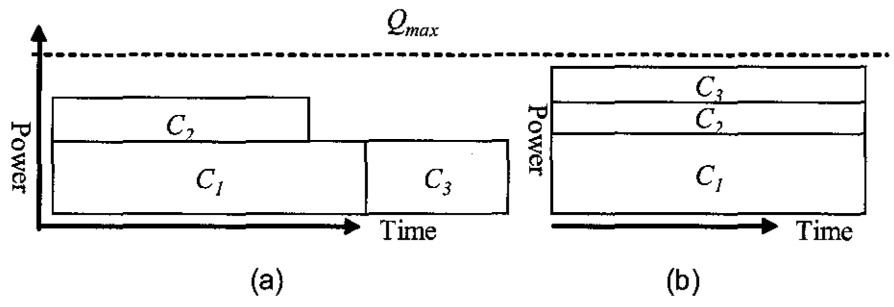


Figure 2.9: Power-constrained scheduling with variable test frequencies allows better power utilization to minimize the test application time.

out of the scope of this chapter. However, since the use of multi-frequency clocks in IC design is commonplace, we assume that such frequency divider implementation is possible, without causing timing violation. In the proposed algorithm (Section 2.6.4), the choice of frequency values are constrained; these values can be selected based on the actual clock frequencies that can be made available on-chip.

2.6.4 Forming Non-Overlapping Test Groups

The distinctive characteristic of our test scheduling methodology (Section 2.6.6) is that it produces a delivery schedule for only a very small subset of test data packets of every core that can be cycled to produce a complete test data delivery schedule. The small delivery schedule means that a small and simple test program is needed to enumerate the start time for the delivery of each packet. The existing functional TAM approach [6] requires that the delivery time of each data packet to all CUT's be individually specified. For large SoC's, the test program (or the control circuit as proposed by [6]) can be very large.

Therefore, when grouping the cores, we utilize a method—forming non-overlapping test groups (Figure 2.10(b))—that supports this novel aspect to ensure that it can be fully exploited. The grouping in Figure 2.10(b), for an SoC with five CUT's, requires two different test packet delivery schedules which start at time t_i, compared to four for Figure 2.10(a).

A test group is formed by scheduling the core with the longest test time first.

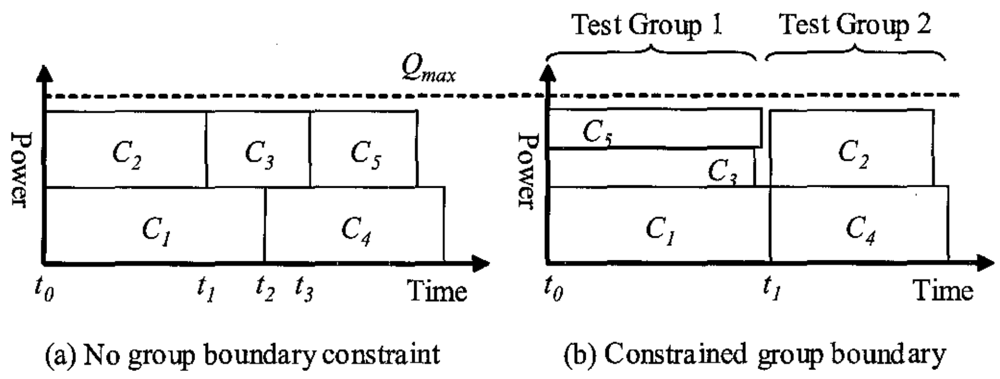


Figure 2.10: Forming non-overlapping test groups by reducing the test frequencies of cores C_3 and C_5.

When scheduling the next core into the same group, its frequency is reassigned to one of the discrete frequencies smaller than the maximum scan frequency. The smallest frequency that will not cause the core test time to exceed the test time of the first core in the group is selected as it meets the twin goals of not exceeding the maximum frequency while approaching it maximally within the preset flip-flop quantity constraint for the clock divider circuit.

When the largest unscheduled core cannot fit the current group within the power constraint, a core that brings total power dissipation for the group closest to the power limit is chosen. This is repeated until no core can fit in, upon which, the same procedure is repeated to create a new group. The process is repeated until all cores are assigned to one of the test groups.

2.6.5 Buffer Sizes

Assuming that each packet starts to be loaded into the scan chains as soon as it arrives at the buffer (i.e. zero buffer loading latency), the required buffer size for a core c_i to store the packet can be specified as [(*packet size in bits*) − (*number of bits loaded into the scan chains during the delivery period of the packet*)], or

$$B_{c_i} = p_{c_i} \cdot w_b - \frac{p_{c_i}}{f_b} \cdot R_{c_i} \tag{2.9}$$

where p_{c_i} = packet size, w_b = bus width, f_b = bus frequency, and R_{c_i} = scan rate.

Equation (2.9) holds under the assumption that the next packet is delivered only when the previous packet has already been scanned in completely. Therefore, the total buffer size, B_{total}, is given by equation (2.10) for all cores $c_i \in M$, where M represents all cores under test in the SoC.

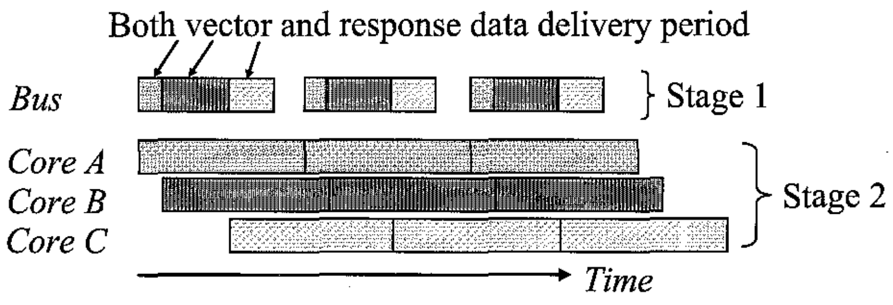
$$B_{total} = \sum_{c_i \in M} B_{c_i} \tag{2.10}$$

2.6.6 PAcket Set Scheduling (PASS) Algorithm

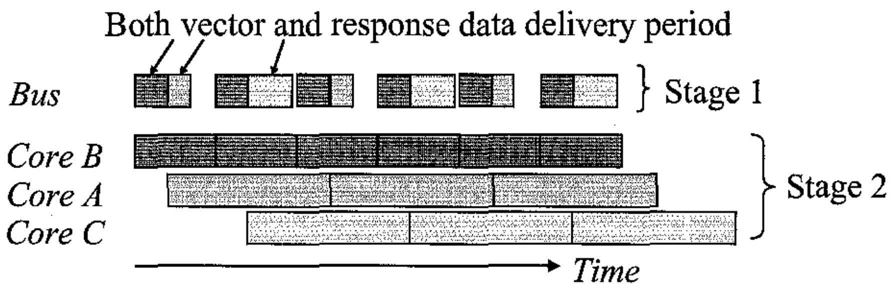
The main objective of the PAcket Set Scheduling (PASS) methodology is to find a repetitive packet delivery sequence on the functional TAM and the corresponding packet sizes for each core (under a given constraint of a maximum total buffer size, B_{max}, for the SoC) that minimize the total test application time. The repetitive delivery sequence is good at simplifying and minimizing the size of the test program because of its looping nature. On the other hand, the limitation of such a delivery sequence is that the job of all members in the repetitive group must be completed before the next (identical) job can be started. In this case, *job* refers to the task of scanning in the test vectors into the scan chains, which must be performed at each CUT. In other words, if there is one member of the group that is late in completing the current job, all other members need to stall while waiting for the slowest member to complete, prior to initiating a new group delivery sequence.

In the scan-based testing perspective, the cores with a smaller number of scan chains (n_s) or those tested at a lower scan frequency (f_{scan}) take a longer time to complete, for the same amount of test data. Therefore, cores with larger $n_s \times f_{scan}$ product require a larger test packet (therefore, larger buffer space) in order for the test application of each packet of each core to have equal scan-in time (Stage 2) as shown in Figure 2.11(a). Equal scan-in time is necessary to avoid stalling.

To reduce the total required buffer size, the packet size for core B can be halved and the delivery sequence changed from A-B-C (Figure 2.11(a)) to B-A-B-C (Figure 2.11(b)). Both Figures 2.11(a) and 2.11(b) show three repetitions of the



(a) Variable packet sizes for test time minimization.



(b) Improved delivery sequence for packet size minimization.

Figure 2.11: Packet size and test time optimization through packet splitting.

smallest subset of delivery sequences A-B-C and B-A-B-C, called *packet sets*, respectively. In Figure 2.11(b), cores A and C are said to belong to the split-1 group because only one packet is delivered in the packet set (A and C only appears once in the B-A-B-C sequence). Similarly, core B is said to belong to the split-2 group; packet B is split into *two* smaller packets from the original packet size equivalent to the split-1 packet. The packet set scheduling problem, Ψ_{PASS}, is defined as follows:

Problem 2.1 (Ψ_{PASS}) Given a test group, G, consisting of n cores $\{c_1, c_2, \ldots, c_n\}$, the characteristics and test requirements for each core c_i, and the selected test frequency for each core in G, determine the smallest repetitive delivery pattern (i.e. packet set) which can be continuously repeated to form a complete test data delivery schedule utilizing the functional TAM, for all cores in G with the following objectives.

1. The total test application time for all the cores in the test group G is minimized.
2. The variations in the buffer sizes of all cores in G are minimized, to ensure that the total test buffer cost is minimized. ■

The basic principle illustrated by the packet splitting example in Figure 2.11 fulfills the two objectives defined by the Ψ_{PASS} problem. To minimize the buffer cost, the most important characteristic is the ratio between the buffer sizes of all cores in the test group. In the best case, if the ratio is one for all cores, then the minimum buffer size can be used for each core. By ensuring there is no gap between the stage 2 operation of each test data packet (Figure 2.11), the TAT is minimized. This can be achieved by selecting the proper delivery sequence for all the cores in the test group.

The packet set scheduling algorithm to solve Ψ_{PASS} problem consists of three steps. First, to determine how to split the test packet for each core (i.e. finding the split ratios) so that the individual packet sizes are equal. If the packet sizes are not equal, the largest packet will become the constraint when minimizing the total buffer sizes as illustrated in Section 2.3. In the second step, once the split ratio has been identified, the packet sizes are determined by solving a set of linear equations. In the third step, a sequence of packet set delivery schedules is systematically formed.

Step 1

Let us consider a test group which has n cores to be tested simultaneously. In the first step of the algorithm, all $k < n$ cores with scan rates smaller than the average scan rate for all cores are considered to have a split ratio of one—the smallest split ratio. This is because other larger cores will be assigned split ratios of larger than or equal to one. Under the PASS scheme, the smallest possible number of packets is desirable when forming a packet set in order to minimize the complexity of the resulting test program. Before proceeding, we define a relevant terminology to aid the description of the algorithm.

Definition 2.9 Assuming that the bus delivery rate is sufficiently high, a packet set is considered to be in *perfect-fit* if

(i) it does not have cores that are waiting for test data,
(ii) there are no two consecutive packets delivered that belong to the same core, and
(iii) the number of packets between adjacent split-1 packets are equal.

Furthermore, all three conditions need still hold when two adjacent perfect-fit packet sets are cascaded, except possibly for the initial or final legs of test application. ■

Figure 2.12 shows a perfect-fit delivery sequence, where the test group consists of nine cores, C_1 to C_9. Between the four split-1 cores ($C_1 - C_4$), eight packets belonging to other cores ($C_5 - C_9$) are delivered (perfect-fit condition (iii)). In order to ensure the perfect-fit criteria are not violated when forming a perfect-fit packet set consisting of split-1 and split-r cores, for any value of $r > 1$ such that $(k \; mod \; r) = 0$, the number of split-r cores must equal $(d \times k/r)$ for some positive integer d. This requirement is imposed in order to achieve an even utilization of bus time, as implied by condition (iii) of Definition 2.9, we need to schedule the same number of split-r packets in between the delivery of split-1 packets. This forms d subgroups of (k/r) split-r cores which make up the split-r group.

To determine the split-r cores, we iteratively check for all possible values of r, starting with the smallest. Let R_{avg} be the average scan rate of split-1 cores. For the remaining cores with split ratio value unassigned, if there exist k/r cores c_i that fulfill $R_{c_i} < \beta r \times R_{avg}$ for some constant β, then all the k/r cores are assigned split ratio values of r. The constant β gives a cut-off limit on the largest core to be assigned to split-r group. It limits the relative buffer sizes of split-r cores and split-1 cores. The value of $\beta = 1.5$ was chosen after thorough experimentation.

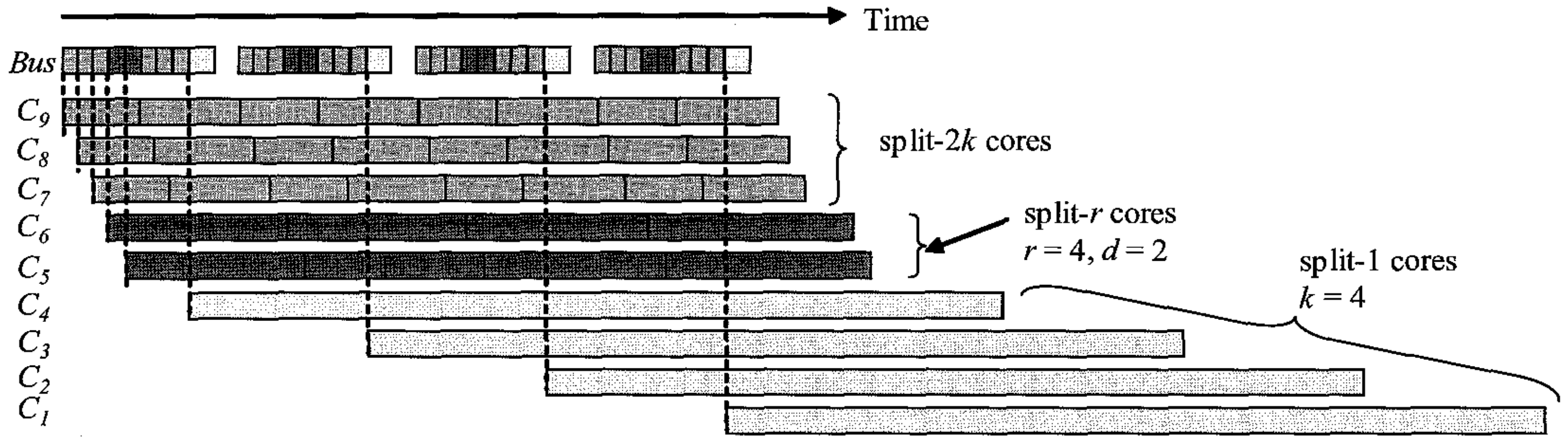


Figure 2.12: Packet set scheduling algorithm. An example of a *perfect-fit* delivery sequence.

The process above is repeated when identifying the next k/r subgroups of split-r cores. As a result, d subgroups of k/r cores are assigned the split ratio of r. If no subgroup could be found for the current value of r, this process is repeated for the next larger value of r until r equals k. Then, the remaining $q = (n - k - d \times k/r)$ cores are assigned a split ratio of $2k$ to form the split-$2k$ group.

Instead of assigning split ratio of $2k$ for the remaining cores, the same procedure for forming the split-r group can be extended to form other split groups. However, to minimize the algorithm complexity, we have chosen only three split ratio values (1, r, and $2k$) since they provide sufficiently good results (i.e. overall test application time) as illustrated in Section 2.8.

The heuristic process described above for solving Ψ_{PASS} can be summarized by the Algorithm 2.1 below, called PAcket Set Scheduling (PASS) algorithm; the algorithm gives a minimum size schedule of a packet-based test data transportation. In the algorithm, a constant k_f is used (lines 4 and 7) to initially separate out the cores which require smaller amount of test data compared to the other cores in the test group. The constant value of $k_f = 1.5$ was chosen through thorough experimentation on several modified benchmark circuits based on the ITC'02 benchmark suite [1].

Algorithm 2.1 PASS (Ψ_{PASS})

```
 1. $VOD$ = Volume of test data in bits;
 2. $VOD_{avg}$ = Average $VOD$ of all modules;
 3. For every core $c_i \in G\{$
 4.    $c_i \in \{small\_modules\}$ if $VOD_{c_i}/VOD_{avg} \leq k_f;\}$
 5. $VOD_{sm}$ = Average $VOD$ of $\{small\_modules;\}$
 6. For every core $c_i \in G\{$
 7.    If $VOD_{c_i}/VOD_{sm} \leq k_f$ {
 8.       $c_i \in$ {split-1 group}}}
 9. $k$ = Number of split-1 group members;
10. Sort the remaining modules in increasing VOD;
11. If ($k$ is a prime number) { /* except $k = 2$ */
12.    If next larger $VOD > VOD_{sm} \times (k+1)/2${
13.       All qualified cores $\in$ split-$k$ group;
14.       Other $q$ cores $\in$ {split-$2k$ group;}
15.       Exit procedure;}
16.    Else {
17.       Remove one split-1 group member whose $VOD$ is farthest from others;
18.       Update $k = k - 1$;}} /* $k$ becomes non-prime */
19. If ($k$ is not a prime number){
20.    $r_t$ = Smallest $r_t > 1$ such that $k \; mod \; r_t = 0$;
21.    $r_{t+1}$ = Next larger $r_t$ such that $k \; mod \; r_{t+1} = 0$;}
22. While ($r_t < k$) {
23.    If $c_i$ is qualified (i.e. $VOD_{c_i}/VOD_{sm} \leq avg(r_t, r_{t+1})$)
24.       If the next $k/r_t - 1$ modules in $G$ also qualified{
25.          All $k/r_t$ qualified modules $\in$ {split-$k$ group;}}
26.       Repeat to find the next $k/r_t$ subset of qualified modules and assign to
          split-$k$ group; /* results in $d$ subsets of $k/r_t$ modules */
27.       All remaining $q$ cores $\in$ {split-$2k$ group;}
28.       Exit procedure;}
29.    Else{
30.       $r_t = r_{t+1}$;
31.       $r_{t+1}$ = Next larger $r_t$ such that $k \; mod \; r_{t+1} = 0$;}
    }                 /* exit While loop if $r_t = k$ */
32. For all remaining unscheduled $c_i${
33.    $ratio = VOD_{c_i}/VOD_{sm}$;
34.    If $ratio$ is closer to $k \times VOD_{sm}$ than $2k \times VOD_{sm}${
35.       Assign $c_i \in$ {split-$k$ group;}}
36.    Else{
37.       Assign $c_i \in$ {split-$2k$ group;}}}
```

Step 2

Once the split ratios are determined, the next step is to determine the packet size for each core. Equation (2.11) describes the scan in time of a test packet, where

w_b = bus bit width, p_{c_i} = packet size, and f_{c_i} = scan frequency, for core c_i.

$$T_{c_i,p} = \frac{w_b \cdot p_{c_i}}{f_{c_i}} \tag{2.11}$$

To preclude introduction of gaps between the test applications of two consecutive packets of a core (condition (*i*) of Definition 2.9) as illustrated by Figure 2.4(b), the packet loading time (i.e. scan in time of a packet worth of test data from the input buffer into the scan chains) multiplied by the corresponding split ratio must be identical as shown by the illustrative example in Figure 2.13. This is necessary to ensure that there are no gaps between the packets within the packet set and between adjacent packet sets.

Equation (2.12) describes the general packet loading times as illustrated by Figure 2.13, where r and $2k$ are the corresponding split ratios for each core. Equation (2.12) assumes that the split-1, split-r, and split-$2k$ cores are labeled C_1 to m_k, m_{k+1} to $m_{k+dk/r}$, and $m_{k+dk/r+1}$ to $m_{k+dk/r+q}$, respectively. For example, given $k = 4, r = 4, d = 2$, and $q = 3$ as in Figure 2.12, cores $\{C_1, C_2, C_3, C_4\} \in$ split-1 group, $\{C_5, C_6\} \in$ split-r group, and $\{C_7, C_8, C_9\} \in$ split-$2k$ group, respectively. Packet size, p_{c_i}, and buffer size, B_{c_i}, for each core c_i can be calculated by solving equations (2.9), (2.10), and (2.12) simultaneously. For each value of B_{total}, a unique solution can be obtained.

$$\begin{aligned}
&\frac{w_b \cdot p_{c_1}}{f_{c_1}} = \cdots = \frac{w_b \cdot p_{c_k}}{f_{c_k}} \\
&= r \cdot \frac{w_b \cdot p_{c_{k+1}}}{f_{c_{k+1}}} = \cdots = r \cdot \frac{w_b \cdot p_{c_{k+\frac{dk}{r}}}}{f_{c_{k+\frac{dk}{r}}}} \\
&= 2k \cdot \frac{w_b \cdot p_{c_{k+\frac{dk}{r}+1}}}{f_{c_{k+\frac{dk}{r}+1}}} = \cdots = 2k \cdot \frac{w_b \cdot p_{c_{k+\frac{dk}{r}+q}}}{f_{c_{k+\frac{dk}{r}+q}}}
\end{aligned} \tag{2.12}$$

Step 3

In step one, the cores are assigned to either split-1, split-r, or split-$2k$ groups. Once the split ratios are determined, the complete packet set schedule that fulfills conditions (*ii*) and (*iii*) of Definition 2.9 can be systematically represented by Figure 2.14, assuming k and q cores for split-1 and split-$2k$ groups, respectively.

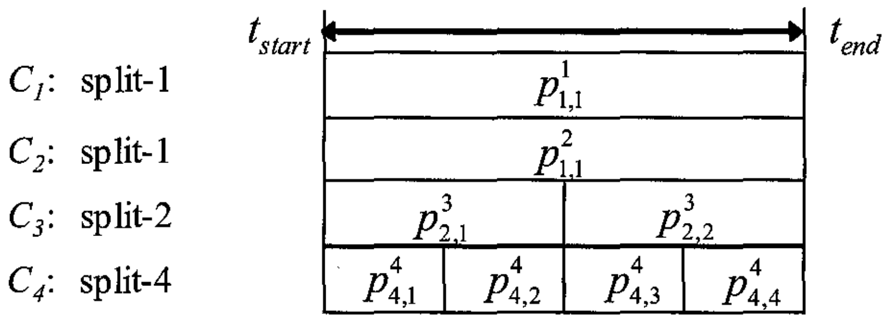


Figure 2.13: Equating split ratio × packet loading time for all cores in a test group.

$$\begin{array}{cccccccc}
p^1_{2k,1} & p^2_{2k,1} & \cdots & p^q_{2k,1} & p^1_{r,1} & p^{1+k/r}_{r,1} & \cdots & p^{1+(d-1)k/r}_{r,1} \\
p^1_{2k,2} & p^2_{2k,2} & \cdots & p^q_{2k,2} & p^1_{1,1} & & & \\
p^1_{2k,3} & p^2_{2k,3} & \cdots & p^q_{2k,3} & p^2_{r,1} & p^{2+k/r}_{r,1} & \cdots & p^{2+(d-1)k/r}_{r,1} \\
p^1_{2k,4} & p^2_{2k,4} & \cdots & p^q_{2k,4} & p^2_{1,1} & & & \\
\vdots & \vdots & & \vdots & \vdots & & & \\
p^1_{2k,2k-1} & p^2_{2k,2k-1} & \cdots & p^q_{2k,2k-1} & p^{k/r}_{r,r} & p^{k/r+k/r}_{r,r} & \cdots & p^{k/r+(d-1)k/r}_{r,r} \\
p^1_{2k,2k} & p^2_{2k,2k} & \cdots & p^q_{2k,2k} & p^k_{1,1} & & &
\end{array}$$

Figure 2.14: Packet set delivery sequence for a test group with three split groups—split-1, split-r, and split-$2k$.

Each $p^c_{i,j}$ represents a test packet delivery followed by a response packet retrieval where,

c = core number from split-i group

i = split ratio for core c

j = packet number for core c, and $j \leq i$

In Figure 2.14, the odd rows (horizontal) show the schedule delivery for q split-$2k$ packets followed by d split-r packets. The even rows show the schedule delivery for the subsequent q split-$2k$ packets followed by a single split-1 packet from one of the k cores. The packet delivery ordering is from left to right and top to bottom.

The retrieval of the response packet is scheduled after every test packet delivery. This can be implemented using the write-read data transfer which requires a single address cycle. If it is not supported by the functional operation, an additional address cycle would be required to initiate the read cycle to fetch the response packet. Another possible approach is to introduce a special write-read instruction specifically for the test data transportation—i.e test stimuli write followed by test response read. Both these approaches require minimal overhead on the control algorithm.

2.7 Advantages of the Buffer-based Wrapper

We have added local buffers to each core-under-test in order to enable concurrent core tests. Furthermore, when doing so the buffer provides clock isolation between the core and the bus interface. This is an intrinsic property of the buffering mechanism.

In the case of a dedicated test access mechanism, the external tester port connects directly to the TAM wire, which in turn connects directly to the wrapper scan chains. As a result, the scan clock is physically tied to the tester clock. If the maximum tester frequency is 200 MHz, then the scan frequency, the tester frequency, and the TAM frequency cannot exceed 200 MHz and must be identical. Similarly, if the maximum scan frequency is 300 MHz, the tester frequency must operate at 300 MHz or less, whichever is smaller.

With the buffered wrapper, the scan frequency is independent of the bus frequency. With the advanced SoC's, the bus frequency is most likely to be higher than the scan frequency. Therefore, in the experimental results (Section 2.8), we also reported the case when the bus frequency is twice the scan frequency. This is to illustrate the advantage of the proposed approach compared to the dedicated TAM approach.

2.7.1 Bandwidth Matching with Low Speed Testers

In order for the higher functional bus speed advantage by means of clock decoupling to translate into real benefits, the larger bandwidth required must be sustainable from the external tester. Most testers however are not capable of operating at the higher functional bus frequency, with the exception of maybe a few very advanced and expensive testers. Without the tester support, the advantages of increasing the bus frequency cannot be immediately realized.

These testers normally comes with a large number of probe ports, some ranging in the thousands. By using a simple bandwidth matching circuitry at the chip's I/O port as illustrated in Figure 2.15. I/O port bandwidth and the ATE bandwidth are given in equations (2.13) and (2.14), where n_{port}, f_{bus}, n_{probe}, and f_{probe} are the I/O port bit width, functional bus frequency, number of ATE probes, and probe's operating frequency.

$$BW_{port} = n_{port} \times f_{bus} \tag{2.13}$$

$$BW_{ATE} = n_{probe} \times f_{probe} \tag{2.14}$$

Figure 2.15 illustrates that with $n_{probe} = 4 \times n_{port}$, the ATE can sustain the bandwidth required by the functional bus operating at four times its probe's frequency. Using this bandwidth matching scheme, we can take advantage of the buffer-based wrapper.

2.8 Experimental Results

We have conducted experiments on several ITC'02 benchmark [1] circuits in order to verify the efficiency of the proposed algorithm. Since the power dissipation information is not available (except for h953 circuit) in the benchmark suite definition, we obtained power information for p93791 and p22810 from [19] and d695 from [13]. In order to analyze the efficiency of functional TAM utilization for test data delivery, a single shared functional bus is assumed to be connected to every core.

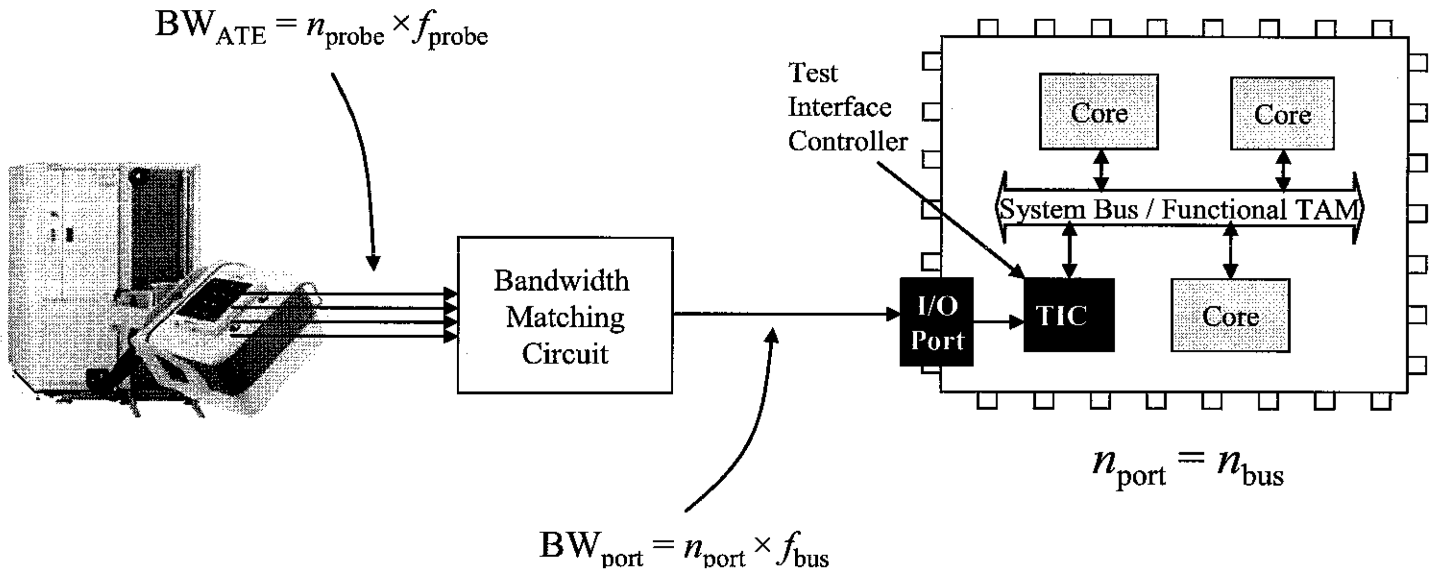

$BW_{ATE} = n_{probe} \times f_{probe}$
Bandwidth Matching Circuit
Test Interface Controller
Core
Core
System Bus / Functional TAM
I/O Port
TIC
Core
$n_{port} = n_{bus}$
$BW_{port} = n_{port} \times f_{bus}$

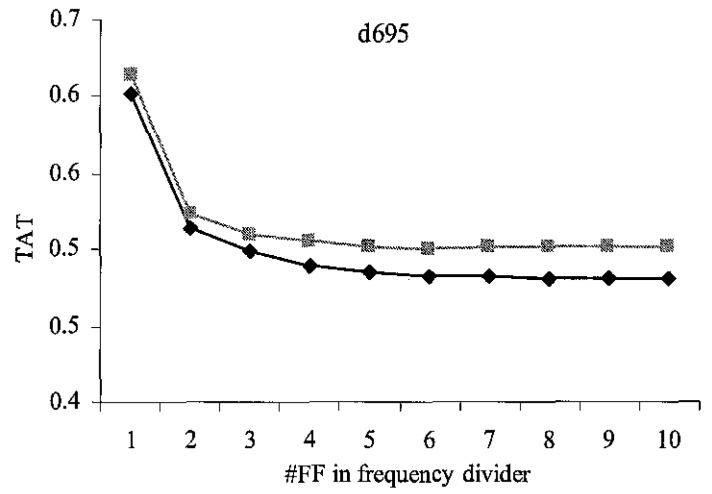
Figure 2.15: Bandwidth matching mechanism to increase the tester's effective bandwidth.

Table 2.1: Selected benchmark circuits from the ITC'02 benchmark suite. Power information for p93791 and p22810 is from [19], and d695 from [13].

Module	p93791	p22810	d695
1	7014	1238	660
2	74	80	602
3	69	64	823
4	225	112	275
5	248	2489	690
6	6150	144	354
7	41	148	530
8	41	52	753
9	77	2505	641
10	395	289	1144
11	862	739	
12	4634	848	
13	9741	487	
14	9741	115	
15	78	580	
16	201	237	
17	6674	442	
18	113	441	
19	5252	167	
20	7670	318	
21	113	1309	
22	76	260	
23	7844	363	
24	21	311	
25	45	2512	
26	76	2921	
27	3135	413	
28	159	508	
29	6756		
30	77		
31	218		
32	396		

The effect of the frequency divider resolution on the test application time is shown in Figure 2.16. In each plot, the bottom curve represents the test application time after the test group has been formed under a power constraint. The higher frequency divider resolution allows us to achieve a shorter test application time. A significant reduction in test time can be achieved within the first four bits of clock divider resolution. The packet scheduling test application time (top curves) are always higher as they incur an additional overhead when splitting the test data into smaller packets.

In order to evaluate the performance of our test scheme, we need to compare with dedicated TAM-based test scheduling approaches. No direct comparison can be offered with previous functional TAM-based test schemes as the experimental



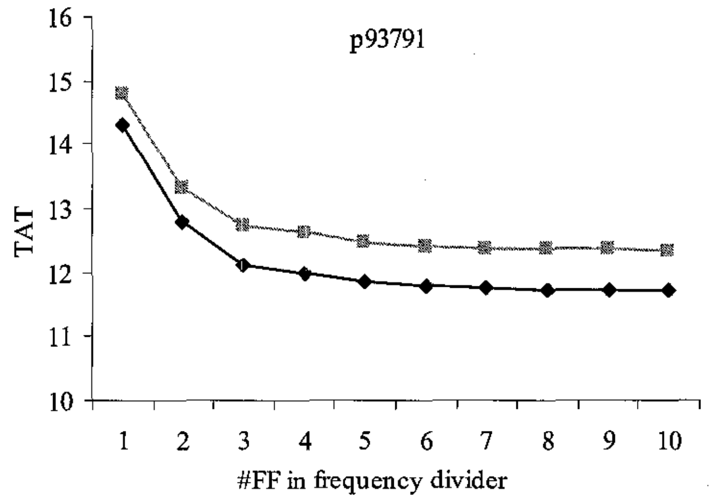


Figure 2.16: Test application time vs. cost of the frequency divider (in flip-flop count) before (solid) and after (dotted) splitting the test data into packets.

results in [6] use four benchmark circuits which lack required comparison information such as information on test data and scan chain configurations that are needed. The authors of [6] subsequently proposed a method which uses a combination of a functional TAM and a dedicated TAM [27], which is subsequently analyzed and compared.

Table 2.2 shows the frequency information for dedicated TAM approaches and two variations of our approaches, *PASSa* and *PASSb*, with distinct bus frequencies. The scan frequency, f_s, is set to the assumed maximum, $F_s = 100$ MHz; therefore all the dedicated TAM-based TAT's [13, 17–19] are divided by 10^5 to convert from the number of clock cycles to time (millisecond). The bus frequency, f_b, for PASSb is double that of dedicated TAM-based and PASSa approaches (but less than the maximum bus operating frequency, F_b) to illustrate the benefit of our buffer-based approach.

In Table 2.3, the TAT's for [13, 17, 18] are all equal at three Q_{max} values for h953 circuit. In our approach, relatively similar results were obtained. No noticeable improvement was achieved when increasing the bus frequency (PASSb). These steady results were due to a single dominant core, C_1, that constrains the TAT minimization for this circuit.

Figure 2.17 shows plots of the TAT for different bus widths. For 64- to 128-bit bus, the TAT is constrained by the largest core; therefore, increasing bus widths has no significant effect on test application time. However, for bus widths between 12 and 48 bits, PASSa delivers improvements of 4.8% and 18.2% over [19] for both maximum power, Q_{max}, values of 3,000 and 10,000 for p22810. PASSb is improved by 25.9% to 47.8% when test data delivery time is the limiting factor. Similar trends can be observed for p93791 in Figure 2.17(c) and Figure 2.17(d). In fact, our test methodology delivers marked improvements in reducing test application

Table 2.2: Experimental setup. Same frequency settings for dedicated TAM-based approaches and our PASSa approach, while PASSb uses higher bus frequency.

	Scan frequency	Bus frequency
Dedicated TAM-based	$f_s = F_s$	$f_b = f_s \leq F_b$
PASSa	$f_s = F_s$	$f_b = f_s \leq F_b$
PASSb	$f_s = F_s$	$f_b = 2 \times f_s \leq F_b$

Table 2.3: Test application time (h953). $B_{total} \leq 100 \times w_b$.

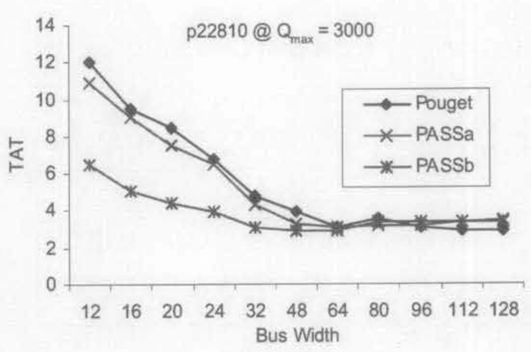
	Test Application Time (ms)				
Q_{max}	[13]	[17]	[19]	PASSa	PASSb
6×10^9	1.22636	1.22636	1.22636	1.21633	1.21942
7×10^9	1.19357	1.19357	-	1.21633	1.22314
8×10^9	1.19357	-	1.19357	1.23164	1.21376

time for smaller bus widths.

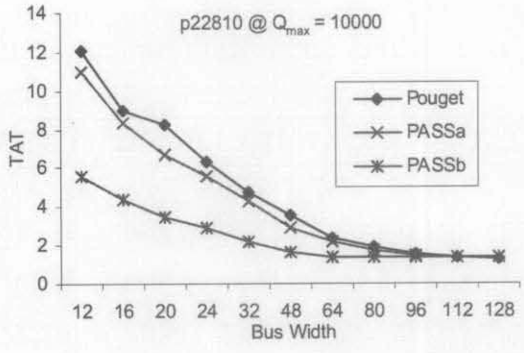
For d695 (Table 2.4), our approach proves to be highly effective, even for the same bus frequency as [13, 19], at all power levels for bus widths ranging from 32 to 80 bits. For 96-bit and wider buses, our methodology though fails to perform as well. It is interesting to note, however, that the dedicated TAM-based approach requires quite elevated levels of TAM overhead in order to outperform our packet scheduling approach using the functional TAM.

In Table 2.5, some performance comparisons with several scheduling approaches is given. The second row shows the TAT for the dedicated TAM-based scheduling without considering power and hierarchy constraints [21]. The TAT is bounded by the lower bound of $T_{LB} = 10.2ms$ [21]. When the design hierarchy is considered as a constraint, the resulting TAT [19] is shown on the third row. On rows 4-6, the TAT of a test set sharing and broadcasting approach is given. When using only the functional TAM (fourth row), the TAT is 27% higher than the case of using only the dedicated TAM [19].

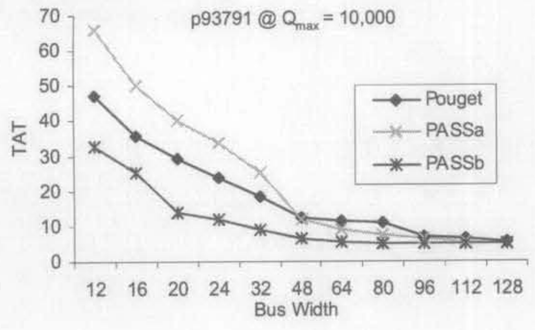
The author improves the performance by using a hybrid architecture with twice the previous bit width (rows 5-6) [27], but comparable hardware overhead due to the same bit width of dedicated TAM as [19]. The last row shows our proposed approach which uses only the functional TAM, with comparable performance when the functional TAM frequency is constrained by the scan frequency. The real advantage is illustrated when a higher functional TAM frequency (not exceeding the maximum functional frequency) is used for test data transportation, which is made possible by the frequency decoupling provided by the buffer architecture.



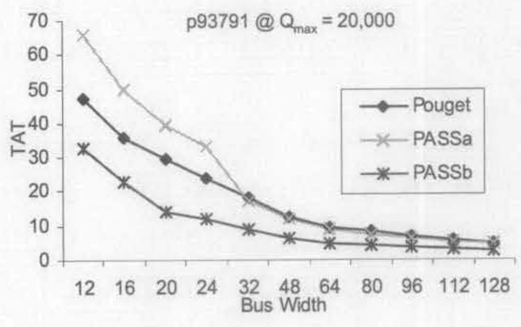
(a)



(b)



(c)



(d)

Figure 2.17: Test application time (ms) vs. bus width for selected Q_{max}. $B_{total} \leq 200 \times w_b$. "Pouget" refers to a dedicated TAM approach proposed in [19].

Figure 2.18 shows the trend in TAT under different buffer size utilization for the two circuits with the same power constraints as in Figure 2.17. The buffer size represents the total size, in multiples of bus width, allocated to all cores in the circuit. It is interesting to note that increasing buffer size only reduces the TAT marginally. Therefore, buffer size can be reduced with only a small

Table 2.4: Test application time (ms) of d695. $B_{total} \leq 50 \times w_b$.

Q_{max}	[13]	[19]	PASSa	PASSb	[13]	[19]	PASSa	PASSb
	Bus width, bw = 32				bw = 48			
1500	0.456	0.435	0.390	0.342	0.310	0.327	0.393	0.325
1800	0.443	0.425	0.390	0.206	0.299	0.321	0.261	0.190
2000	0.432	0.425	0.402	0.233	0.294	0.291	0.294	0.212
2500	0.432	0.418	0.402	0.206	0.290	0.291	0.276	0.153
	bw = 64				bw = 80			
1500	0.276	0.270	0.322	0.327	0.209	0.244	0.329	0.327
1800	0.245	0.239	0.207	0.196	0.205	0.188	0.192	0.191
2000	0.242	0.219	0.234	0.213	0.192	0.187	0.224	0.213
2500	0.237	0.219	0.206	0.153	0.192	0.187	0.173	0.153
	bw = 96				bw = 112			
1500	0.209	0.234	0.324	0.324	0.168	0.194	0.328	0.328
1800	0.181	0.188	0.190	0.189	0.150	0.188	0.192	0.191
2000	0.178	0.175	0.211	0.210	0.141	0.146	0.214	0.214
2500	0.158	0.173	0.153	0.152	0.141	0.140	0.147	0.152
	bw = 128							
1500	0.168	0.194	0.321	0.321				
1800	0.149	0.168	0.197	0.189				
2000	0.141	0.145	0.208	0.208				
2500	0.130	0.134	0.160	0.160				

penalty on TAT. Table 2.6 shows the area cost in terms of average buffer sizes per core, averaged over $w_b = 32...128$, for the corresponding TAT results reported in Figure 2.17 and Table 2.4.

With the flexibility of bus frequency selections, unique to our proposed approach as a dedicated TAM-based approach is unable to utilize such flexibility, we can further improve the TAT while ensuring that nothing more than minimal bus widths are utilized. This is illustrated by PASSb in Figure 2.17 and Table 2.4.

Figure 2.19 shows the trend in the test application time for several bus fre-

Table 2.5: Performance comparison of several testing approaches (d695) [$^*Q_{max}$=2500; maximum hardware overhead constraint = 300† and 400‡ units].

References	TAT (ms)	Configurations
Unconstrained [68]	11	64-bit TAM
Hierarchy Constraints [19]	21.9* 20.5	64-bit TAM
Test Set Sharing and Broadcasting [27]	26.1	64-bit functional bus
	20.4† 18.5‡	64-bit functional bus and 64-bit TAM (Total = 128 bits)
Proposed	18.8* 13.5*	64-bit functional bus, $f_b = f_s$ 64-bit functional bus, $f_b = 2 * f_s$

Table 2.6: Average buffer sizes per core for the corresponding TAT of PASSa and PASSb in Figure 2.17 and Table 2.4.

Circuit	p22810		p93791		d695			
Q_{max}	3000	10000	10000	20000	1500	1800	2000	2500
PASSa	6.66	6.39	3.89	4.31	4.45	3.75	4.00	4.06
PASSb	5.95	6.40	5.03	5.23	4.71	3.78	3.87	3.87

quency, f_{bus}, values in the range of $0.5 \times f_{scan}$ and $3 \times f_{scan}$. In both cases of Q_{max} = 2000 and 2500, the TAT for 32-bit functional TAM matches that of the 64-bit functional TAM when the test frequency is about 2.5 times the scan frequency. This is because, at this (or larger) bus frequency, the test application time is constrained only by the maximum speed of the scan operation. There are two major advantages of the proposed approach, which utilizes the functional TAM. First, the TAT can be reduced without increasing the TAM area cost. Second, the frequency increase is for the functional bus and not for the scan operation, thus retaining the scan power unaffected. This is a very useful characteristic of the functional bus reuse test scheme because the increased bus capacity in the

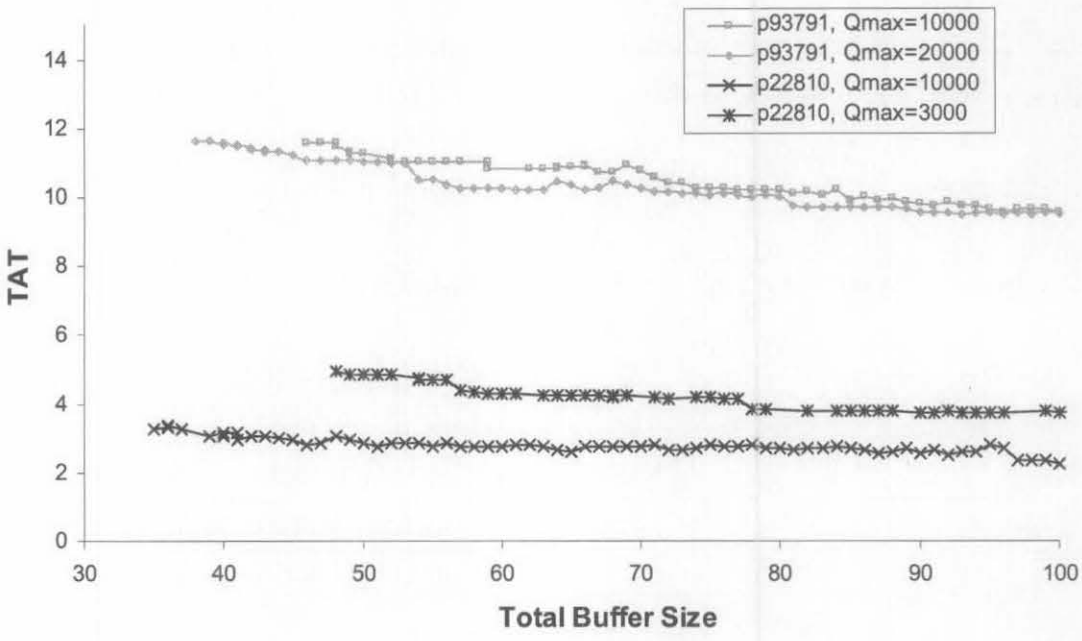


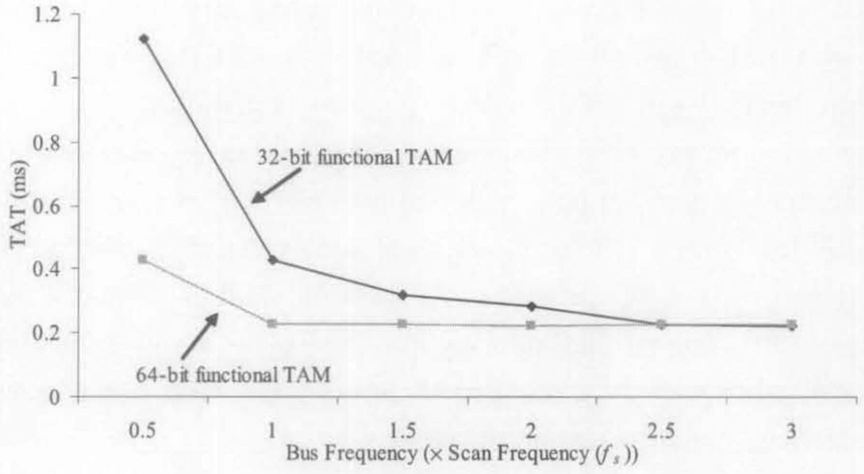
Figure 2.18: Little reduction in the test application time (ms) as the total buffer sizes ($\times w_b$ flip-flops) are increased. Bus width, $w_b = 32$ bits.

advanced SoC's, which this method can benefit from.

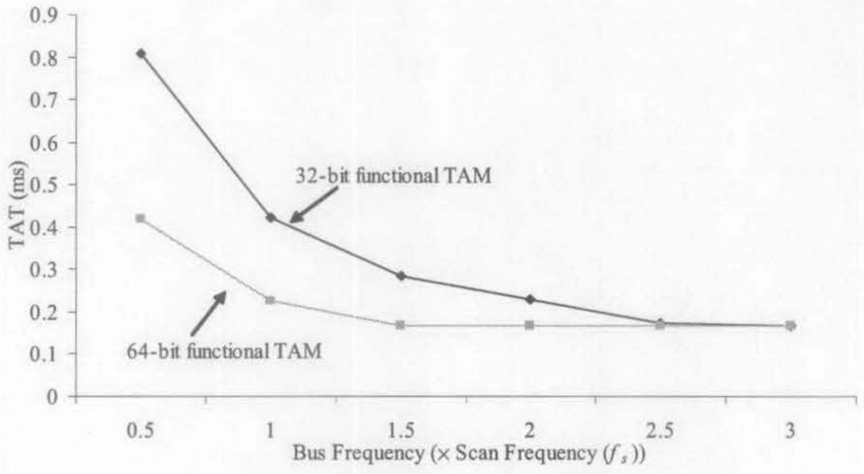
2.9 Conclusion

The functional bus is an essential part of any SoC design. As the performance of SoC's increases, the bus speed has also been increased even though not at the same pace. There are many different bus architectures and protocols that provide various functionalities and performance. Therefore, this chapter explains our research effort to take advantage of the existing functional bus for testing purposes.

The utilization of the functional bus for power-constrained core-based SoC testing entails a number of challenges. These include frequency and bit-width mismatch between the bus and the cores under test, allocation of bus time slots



(a) $Q_{max} = 2,000$



(b) $Q_{max} = 2,500$

Figure 2.19: Reducing the test application time of d695h1 benchmark circuit by using a functional bus frequency larger than the scan frequency (f_{scan}), without increasing the scan power during test.

for an efficient test data delivery schedule that maximizes bus utilization and that ensures that all cores always have the test data that they need to continue testing simultaneously without exceeding the power constraint.

We have herein proposed an efficient methodology that overcomes all of these challenges through a test support architecture design framework and an algorithmic design framework. The proposed methodology offers a solution that also minimizes the size of the test program. The experimental data clearly showcases the benefits of the proposed methodology in reducing test application time especially for smaller bus widths, while also eliminating the need to add extraneous TAM's to the SoC solely for testing purposes.

Chapter 3

Test Scheduling for Multiprocessor SoC's with Hierarchical Buses

3.1 Introduction

Designs with multiple embedded processors are common in today's high performance systems. This type of system-on-chip (SoC) necessitates the use of high capacity on-chip interconnects such as hierarchical buses and networks-on-chip (NoC). From the testing perspective, the best-adopted core-based test approach for SoC's is based on the use of test wrappers [3, 9, 69–71] that allow the cores to be isolated during the test phase, enabling test reuse for the embedded cores. In tandem, a Test Access Mechanism (TAM) is added to provide delivery and control paths to each core from an external automatic test equipment (ATE). The test requirements and challenges of these core-based designs have also been discussed in [8].

Several variations of TAM's have been proposed. However, the most commonly used TAM is similar to a functional bus, but requires introduction of a separate bundle of physical wires—a *dedicated TAM* [9, 72]. With regards to the dedicated TAM approach, numerous test scheduling methods have been proposed [6, 12–50] and briefly discussed in Chapter 2. All these test scheduling and optimization works rely on the introduction of a dedicated and extraneous TAM for the test data transportation.

The benefits of a dedicated TAM are dramatically diminishing as the resources of communication networks in the current and emerging SoC's become abundant [73]. Not only does the added TAM cost additional area, but more importantly, it also adds additional complexity to the already high routing congestion of the functionally required short and long distance interconnects such as the functional buses. The TAM wires are especially costly in the chip fabrication process with feature sizes of 90-nm and smaller. The cost of TAM wires can be minimized by connecting cores to its adjacently-placed cores. Nevertheless, this wire-length-optimized TAM architecture could constraint the test scheduling, thus preventing potential test time reduction. Co-optimization methods as proposed in [43, 47] could alleviate this problem. However, since the functional interconnects are readily available on-chip, they could be used in place of dedicated TAM's—a *functional TAM*, thus eliminating the unnecessary TAM area cost. Since functional interconnects are designed and optimized with the functional operation in mind, their adoption as functional TAM's is not as straightforward and flexible as that of dedicated TAM's.

Several groups propose functional TAM methods discussed in Section 2.2. In addition, we have proposed a wrapper architecture and a packet-level test scheduling methodology specifically targeting the reuse of a flat bus in Chapter 2. These methods [6, 23–26], however do not consider explicitly SoC's with hierarchical buses and multiple embedded processors. Therefore, we are proposing a test data transportation methodology which utilizes the existing hierarchical functional buses; in addition, the proposed method leverages on the processing power of the existing embedded processors for test data generation. Since the buses provide direct access to the embedded cores, the intrusive methods of making cores transparent are also avoided.

In our proposed method covered in this chapter [74], we explicitly schedule the transportation of data packets through hierarchical functional buses that carry the test vectors and responses between embedded processors and CUT's. The test application times are obtained through complete packet-level simulations of the test data transportation. The proposed approach is unique because of the explicit packet transportation schedule, in addition to the ability to make use of the hierarchical buses and multiple embedded processors.

We begin in Section 3.3 with a delineation of the research scope undertaken in this chapter. The buffer-based test architecture is explained in Section 3.4. In Sections 3.5 and 3.6, the test scheduling methodology based on a hierarchical functional TAM is elaborated. Section 3.7 reports the experimental results on selected benchmark circuits. Finally, concluding remarks are offered in Section 3.8.

3.2 Model of Multiprocessor System-on-Chips (MPSoC)

In order to achieve maximum benefit from a multi-processor system, the best possible communication architecture between the processors, the memories, and the peripherals needs to be used. The choice of communication architecture for the system is critical. Under the constricted scope, the hierarchical bus based Multi-Processor System-on-Chip (MPSoC) model is as illustrated in 3.1. The p93791h1 SoC is based on the p93791 benchmark circuit from the ITC'02 benchmark suite [1], with the functional interconnects and several processor cores added. The p93791h1 circuit consists of various functional cores labeled C_1 to C_{14}. The cores are interfaced through hierarchical buses b_0, b_1 and b_2. Buses b_1 and b_2 are interfaced to bus b_0 through buffered bridges B_1 and B_2, respectively.

A buffered bridge is a store-and-forward bridge that is capable of temporarily buffering an incoming data packet before forwarding it to the other side of the bridge. Accordingly, the buffered bridge provides an interface between the buses while at the same time isolating them, letting through only the necessary traffic. The buffered bridge in reference is also referred to as non-transparent bridge.

The MPSoC cores in Figure 3.1 are grouped into three bus regions—identified by the buses b_0, b_1, and b_2. The first bus region, b_0, consists of eight cores and a processor P_1. All the cores in this bus region—cores C_1, C_2, C_3, C_5, C_6, C_7, C_{12}, and C_{14}—are said to be local to processor P_1 since P_1 can communicate with all these cores directly without relying on other intermediate bus bridges. The second bus region, b_2, consists of three local cores—C_4, C_9, and C_{10}—and a cluster of processors P_2 to P_n. The interconnection within the processor cluster is not illustrated in this model. Similarly, cores C_4, C_9, and C_{10} are said to be local

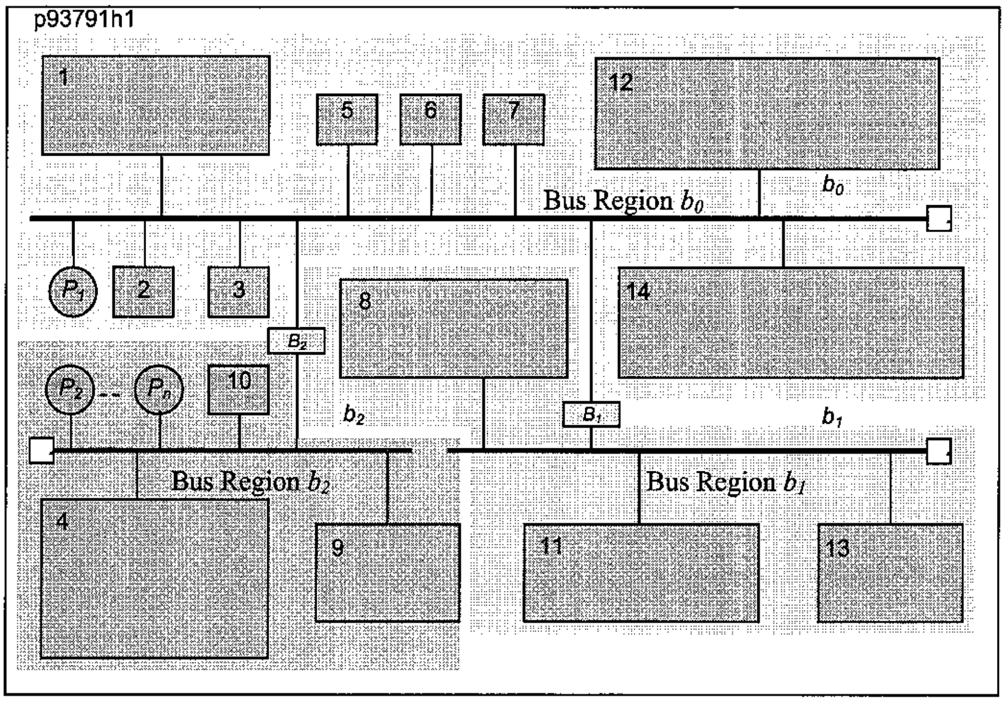


Figure 3.1: General MPSOC architecture.

to processor cluster P_2 to P_n. Bridge B_2 provides an isolation (non-transparent interface) between the b_0 and the b_2 bus regions.

The third bus region, b_1, consists of three cores—C_8, C_{11}, and C_{13}—but no local processor. For generality, we can consider two possible cases. First, if the bridge B_1 is transparent, then bus region b_1 can be merged with bus region b_0. In this case, the local processor to cores C_8, C_{11}, and C_{13} would be P_1. The p93791h1 MPSoC can be said to have only two bus regions, b_0 and b_2.

If the bridge B_1 is non-transparent, then there is no local processor to cores C_8, C_{11}, and C_{13}. Processors from other bus regions can communicate to cores C_8, C_{11} and C_{13} by means of the bridge B_1.

3.2.1 Data Transfer Between Cores and Processors

In the case of local processor communication, the processor first arbitrates for the bus ownership. Once the bus ownership is granted, the processor can broadcast the destination address on the address bus, data on the data bus (if it is a write cycle), and enable the necessary control signals. For a read cycle, the requested data will be delivered by the local cores after n clock cycles. The variable n depends on the implementation of the read cycle timing.

In the case of a non-local processor writing to a core (e.g. P_1 to C_8), the sequence of events are as follows:

1. P_1 arbitrates for bus b_0.
2. Once the bus is granted, P_1 writes to the buffer of bridge B_1.
3. Once a complete data packet is received, B_1 acknowledges to P_1 that it will now proceed to deliver the data to the destination (split transfer mechanism)
4. B_1 arbitrates for bus b_1.
5. Once the bus is granted, B_1 writes to C_8.

After step (3), bus b_0 can be released by P_1 and P_1 subsequently becomes free to perform other tasks.

3.3 Scope

In Chapter 2, a buffer-based test architecture similar to [6] for core-based test application utilizing the embedded processor and the shared functional bus has been proposed. The test application time is minimized by optimizing the bus sharing between multiple CUT's using a novel scheduling methodology called *PAcket Set Scheduling (PASS)*; concurrent core tests are effectively implemented by time-multiplexing the transmission of each data packet carrying the test vectors and responses. The proposed method explicitly schedules the transmission for each data packet that transfers all the test vectors and responses between an embedded test processor and the CUT's. The applicability of the proposed methodology is, however, limited to flat bus and single processor SoC architectures.

In this chapter, we will describe our *Integrated PAcket Set Scheduling (IPASS)* methodology targeting the hierarchical bus based multiprocessor SoC's. Scheduling core tests for a hierarchical bus and multiprocessor SoC involves the tasks of distributing the core tests to multiple processors, and allocating the time slots on the shared functional buses for the delivery of the test data to each CUT. The pro-

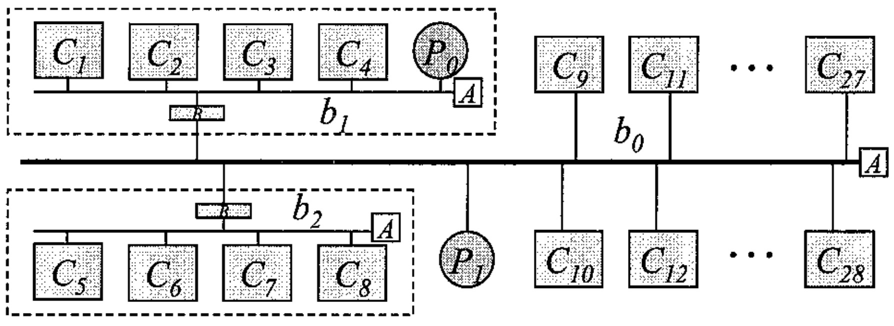


Figure 3.2: General architecture of the MPSoC's considered in this chapter. This example circuit is based on the p22810 SoC circuit from the ITC'02 benchmark suite [1].

posed *power-constrained* and *MultiProcessor PAcket Set Scheduling (MPPASS)* methodologies address the above issues in order to produce an efficient test data delivery schedule.

The design of a multiprocessor SoC (MPSoC) can be implemented using a wide range of architectures depending on the exact design specifications. In addition, various test strategies can be adopted for each embedded IP core. In order to develop an effective SoC test scheduling methodology, a constricted scope of SoC architecture and test requirements are considered in this chapter. The proposed test methodology can be used for core-based test scheduling of multiprocessor SoC's with hierarchical bus architecture. An example MPSoC is given in Figure 3.2 consisting of two embedded processors P_0 and P_1, three hierarchical buses b_0, b_1, and b_2 (interfaced by bus bridges B), and twenty-eight logic cores. Access to the buses is regulated by the arbiters A_i.

The embedded processors, which are existing functional blocks, are reused during testing as test pattern generators and as test controllers. In this chapter, a test scheduling problem which utilizes the hierarchical functional buses as functional TAM's is formulated and a solution is offered under the following assumptions:

- Processors and memories are assumed to be fault-free and therefore not the target of testing.

- All CUT's are enhanced with scan-based DfT's.
- The test uses a combination of pseudo-random test vectors and a small number of deterministic test vectors. The pseudo-random test patterns are generated on-chip by the test processors by running a program, which is loaded into the local memory before the test application. The computation overhead is assumed negligible. The small number of deterministic test vectors, which detect random-resistant faults, are loaded from an ATE into the corresponding processor's local memory location before core-testing begins. The compressed pseudo-random test responses and the deterministic test responses are unloaded to the ATE after the test application is completed.
- Each processor has a fault-free local memory with sufficient capacity to store the test programs and the small number of deterministic test patterns.

Each pseudo-random test vector is transported on the functional buses to the CUT and the corresponding test response is returned to the processor, which then compresses the test response into a signature. The transportation time for each test vector and response depends on the bit width of the functional bus. For example, a 5000-bit test vector with a 5000-bit response requires at least 313 clock cycles (excluding address cycles and buffering delays) on a 32-bit data bus. During this period, only a single pseudo-random test vector is to be generated by the processor. Simple pseudo-random vector generation algorithm requires only a few clock cycles to compute.

The deterministic test vectors are required to detect random-resistant faults, in order to increase the fault coverage of the pseudo-random test vectors. In this chapter, the time to load the test program and the deterministic test vectors, and to unload the pseudo-random test signature and the deterministic test responses is assumed negligible.

3.4 Buffer-Based Test Architecture

In [6], the differences between three types of test access architecture are explained—dedicated TAM, functional bus, and functional bus with buffers. The introduction of buffers allows the core tests to be scheduled concurrently utilizing the functional bus. Our proposed buffer-based DFT, similar to [6] is explained in

Chapter 2 and illustrated in Figure 2.6. The proposed DFT works for an n-bit functional bus and a CUT with m scan chains, where n and m can be arbitrary positive integers. The bit-width conversion is achieved by parallel-serial shifting within the buffer, controlled by the test controller. The first n bits of the core's primary inputs (PI's) and primary outputs (PO's) are connected to the data bus. The remaining u PI's and v PO's are connected to other parts of the SoC.

To isolate the cores during testing, each PI/PO is connected to the bus through a boundary cell, similar to the IEEE 1500's [3], which selects either the scan input (dotted line) or the functional input (solid line). The same control signal (T/N) is used for the boundary cells, the buffers, and the multiplexer to switch between test mode and normal mode. Equal length scan chains are formed by cascading PI's and PO's to the internal scan chains (Figure 2.6) such that both the maximum scan-in depth and the maximum scan-out depth are minimized [21, 66]. Compared to a dedicated TAM-based architecture which utilizes the IEEE 1500 wrapper, the proposed buffer architecture incurs an additional single multiplexer delay on the functional output path.

In the testing scheme for MPSoC proposed in this chapter, the test data are generated by the test processor during the test application or preloaded into local memories. The test processor breaks the test vector data into smaller packets and writes the vector packet to the address corresponding to the target CUT. The test vector data are then delivered through the functional bus to the input buffer as a packet (Figure 2.3), utilizing the necessary transmission protocol supported by the functional bus. Regardless of the data format on the bus, after passing through the functional bus protocol interface, decoded bit-level data are transferred to the input buffer (see Figure 2.8(a)). As a result, the data transfer between a test source and the CUT can be performed through transparent read and write transactions. At the core, the test data decoding is handled by the existing functional interface, after which the raw test data are forwarded to the buffer.

The buffer consists of four main components—input register, output register, fall-through stack, and FIFO buffer controller—as shown in Figure 2.8(a), illustrating the input buffer and the corresponding first-in first-out (FIFO) buffer controller. The architecture of the buffer controller has been explained in detail in Section 2.5.2. Here, we will briefly revisit the operation of the buffer-based core wrapper and its controller and the concurrent test operation utilizing the test buffer and the shared functional bus. The understanding of this concept is critical before discussing the test scheduling methodology for multiprocessor SoC's.

The output buffer (not illustrated) has identical structure as the input buffer but with reverse data flow. The input register latches data from the bus, which then brings the data to the lowest empty slot in the fall-through stack, next to the output register. Upon receiving new data, the status signal α enables the FIFO buffer controller to generate the control signals to shift and load the test data into the wrapper scan chains. When the buffer is empty, control signals are disabled, thereby freezing the scan chains. In the test mode, the multiplexed scan clocks allow the scan chains to hold the test data while waiting for the subsequent data packets from the processor to fill the input buffer. This mechanism makes the proposed buffer-based wrapper flexible in terms of timing; it does not require a streaming and jitter-free transmission of test data to ensure test integrity.

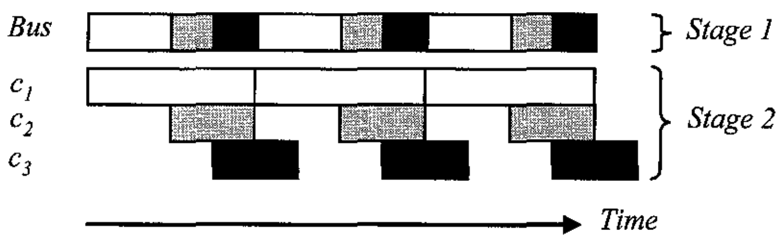
The dotted line (Figure 2.6) shows the path taken by the test vector data from the input buffer into the scan chains. At the same time the test data are being scanned in, the test responses are scanned out of the scan chains into the output buffer. The test responses are then returned to the processor through the same functional path and communication protocol the test vector packet was delivered. To simplify the delivery schedule, the response packets are returned after every successful transmission of a test vector packet (labeled R and V, respectively, in Figure 2.2), except for several initial vector packets prior to the first capture cycle. The response packet empties the local buffer, preparing it for the subsequent vector packet. In order to avoid buffer overflow, the subsequent vector packet is only scheduled for delivery to the core after the successful receipt of the previous response packet by the test processor. This design does not require the buffer size to match the size of every test vector, since the test data can be loaded in batches as they are delivered. The capture signal is generated only after a complete test vector is received and loaded into the scan chains. Therefore, the area cost due to the buffer can be minimized.

The introduction of buffers enables the test application to be scheduled concurrently [6, 54]. Figure 2.3 shows the simplified representation of the buffer architecture for two CUT's excluding the buffer controller. The test data are delivered by an embedded processor to buffers A and B alternately in stage 1 (*data delivery stage*), resulting in a round-robin delivery schedule. In stage 2, the test data in the respective buffers are loaded into the scan chains (*test application stage*) of cores A and B simultaneously. Each vector packet is followed by a confirmation packet in the form of test response data. Figure 2.2 shows the timing diagram of the delivery on the functional TAM (labeled *Bus*) and the test application at each core (labeled *Core A* and *Core B*) assuming the number of scan chains is smaller than the bus width.

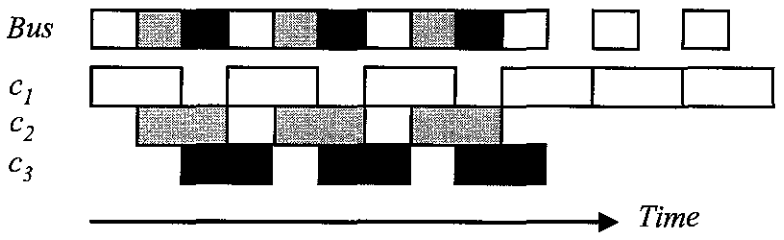
The problems with the round-robin schedule arise when the two CUT's are unequal. In the following illustrative example, three CUT's c_1, c_2 and c_3 with the same number of wrapper scan chains but c_1 has twice the volume of test vector data over the others. The example assumes the bus bandwidth is twice the scan rate. Breaking the data into equal number of packets (Figure 3.3(a)) results in c_2 and c_3 always waiting for test data; test completion times for c_2 and c_3 are delayed. Further optimization is not possible since the bus is always occupied. If the buffer sizes are fixed (Figure 3.3(b)), c_1 requires twice the number of packets over the others. After c_2 and c_3 completes testing, c_1 continues utilizing the bus. Scheduling the test of another core at this time requires a more complex packet delivery schedule. In this chapter, an enhanced version of the round-robin schedule called packet-set-schedule (PASS) is explained in Section 2.6.6, and extended to hierarchical buses architecture in Section 3.6.

3.5 Test Scheduling Methodology

In this section, a test scheduling methodology for the MPSoC cores by reusing the existing flat or hierarchical bus as functional TAM's for the transportation of the test vectors and responses is described. The scheduling methodology consists of two steps. The first step is to decide (*i*) which processors can be best used as test processors, and (*ii*) which cores are to be tested by which test processor. These steps are explained by using the resource graph manipulations (Sections 3.5.1 through 3.5.5) and the test group formation under power constraint (Section 3.5.5) steps. This is followed by the algorithm to determine the optimum



(a) Case 1: Same number of packets



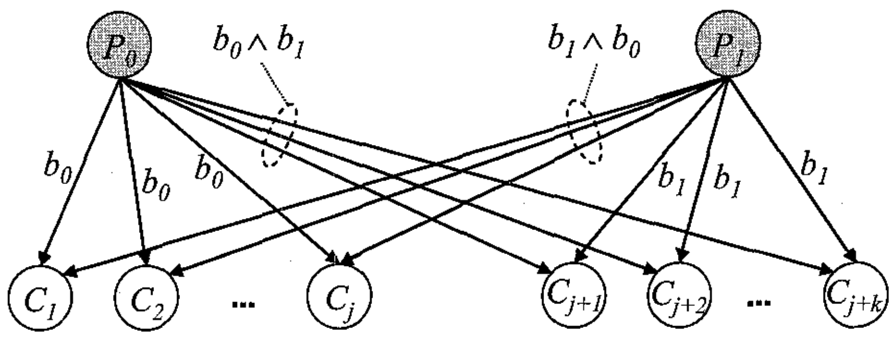
(b) Case 2: Same buffer sizes

Figure 3.3: Illustrative examples of round-robin schedules. For clarity, only vector packets are illustrated. Response packets are excluded from the diagram.

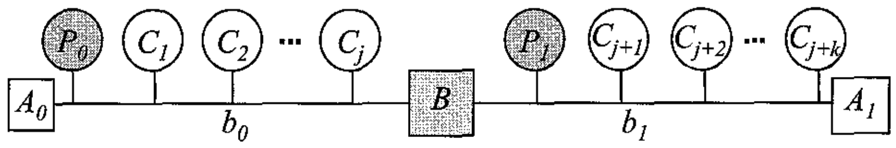
packet delivery schedule for an MPSoC with hierarchical bus (Section 3.6).

3.5.1 Resource Graph

Figure 3.4(a) shows the equivalent resource graph for the example MPSoC in Figure 3.4(b) consisting of two processors P_0 and P_1, two buses b_0 and b_1 (interfaced by a bridge B), and $(j + k)$ logic cores. Access to the buses is regulated by the arbiters A_i. The resource graph provides information about processor-core connectivity. The resource graph captures all the information related to the MPSoC architecture. There is a one-to-one correspondence between the resource graph and the MPSoC interconnection architecture. The node at the beginning of the directed arc represents a bus master-in this case, one of the embedded processors.



(a) Resource graph representation.



(b) Architecture representation.

Figure 3.4: An example hierarchical bus MPSoC.

On the other end of the arc are the cores (bus slaves) which are accessible by the processor.

Functionally, the cores only respond to the processors' read or write requests and do not initiate a bus transaction. The only indirect mechanism for the cores to start a bus transaction is to generate an interrupt signal (in the case of an interrupt-driven design). Based on the priority of the interrupt signal, the processor responds by performing the requested operation at the appropriate time. A typical example is when the cores have data to be delivered to the processor following an earlier read request. The processor's response would be to initiate a read transaction from the core generating the interrupt signal.

In other words, the arcs represent all the physical interconnections between processors and cores. The type of connection shown by each arc is described precisely by the label associated with each arc. A single label (i.e. b_0, and b_1)

indicates a direct connection or a local processor-core interface. A compound label (i.e. $b_1 \wedge b_0$) indicates the presence of non-transparent bridges between the processor and the core. More detailed explanation of each graph can be found in Section 3.5.2.

For example, cores C_1 to C_j are connected to bus b_0 and can be reached by P_0 directly. This is indicated by the b_0 label on the corresponding vertices. The cores can also be reached by P_1 through a bus hierarchy $(b_1 \wedge b_0)$. The AND operator $(\wedge)$ indicates that data delivery passes through a bus bridge.

3.5.2 Test Configuration Graph (TCG)

Definition 3.1 A *test configuration graph* (TCG) is a representation of the physical connectivity between an embedded processor [*test source*] and one or more logic cores [*test sinks*] in an MPSoC. It specifies the test data delivery path between the test source and the test sink(s) on the MPSoC's functional TAM's.

One such test configuration is illustrated by Figure 3.5. In the test configuration graph components diagram, the required on-chip resources are the communication channel (the bus b_u) and the test source (the embedded processor p_q). The test configuration indicates that the processor p_q is required to perform the tests on the core C_i by delivering the necessary test data and retrieving the test responses for analysis.

All possible test configurations can be represented by one of the five test configurations in Figure 3.6 and Figure 3.7. From the resource graph (Figure 3.4(a)) and the test requirements of each CUT, all the test configurations can be extracted and specified using these five basic test configurations or the combinations of multiple of these basic test configurations.

Figure 3.6 illustrates three test configurations (top figures) with their equivalent bus architecture (bottom figures). For each TCG, P_q is the test source, and C_i and C_j are the CUT's. Type I TCG is a test configuration using a local processor. Type II TCG is a hierarchical test delivery architecture, where test data is delivered in two stages as described in Section 3.2.1. Type III TCG is a test configuration for a multi-port processor where both ports are connected to a CUT in parallel by two isolated buses.

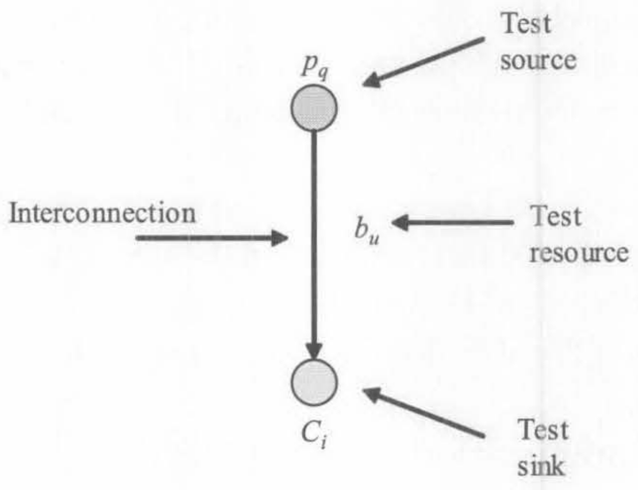


Figure 3.5: Components of a resource graph.

Figure 3.7 shows two possible broadcast-based test configurations. Type IV TCG is a test configuration when the identical CUT's are located within the same bus region, therefore the distance (number of required bus transactions to deliver test data from the processor to the CUT) from the processor to either of the CUT's are identical. Type V TCG is a test configuration when the identical CUT's are on different bus regions. These configurations are also called balanced broadcast test configurations. The broadcast-based test configuration of Type V is possible under the assumption that the processor is capable of delivering the same data on multiple ports within the same clock cycle.

Other more complex test configurations can be constructed by combining and merging these five basic test configurations. For example, the resource graph in Figure 3.4(a) can be decomposed into $2 \times (j + k)$ TCG's of Type I and Type II, where each core has two TCG's.

Any broadcast test configurations can be considered can be classified as either balanced or unbalanced broadcast depending on the relative distances between the processor and each of the CUT's in terms of the number of bus transactions. In Section 3.5.5 we explain how these TCG's are used in the power-constrained scheduling.

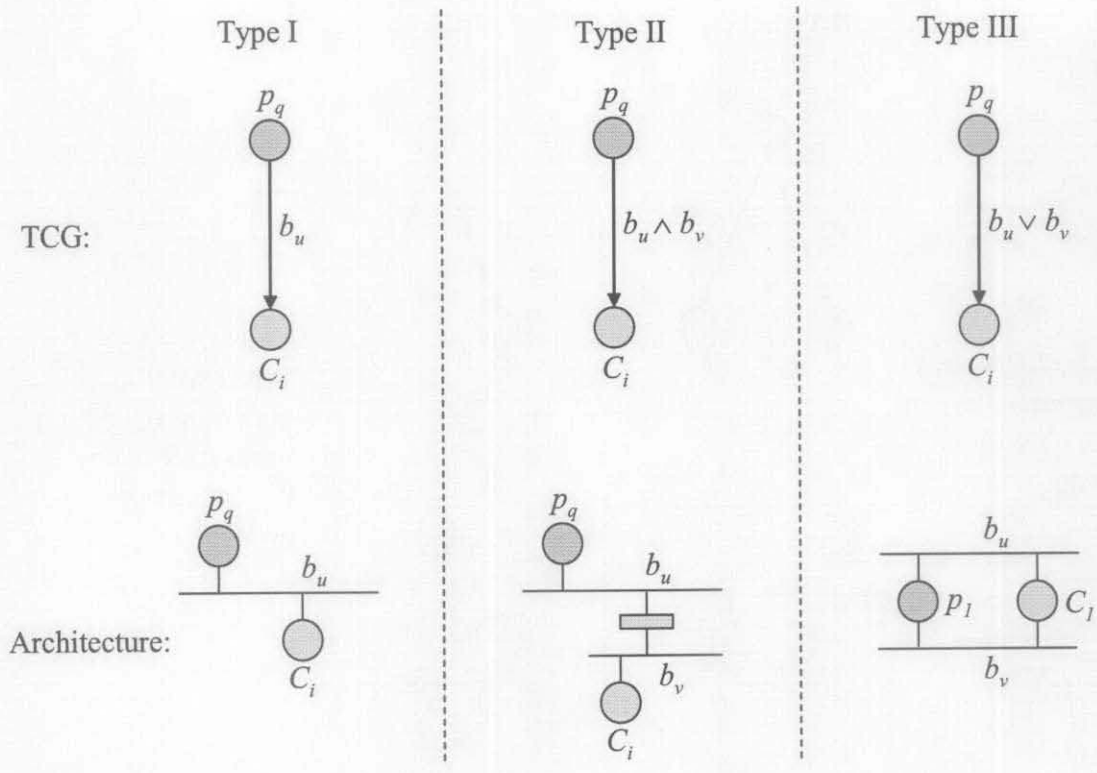
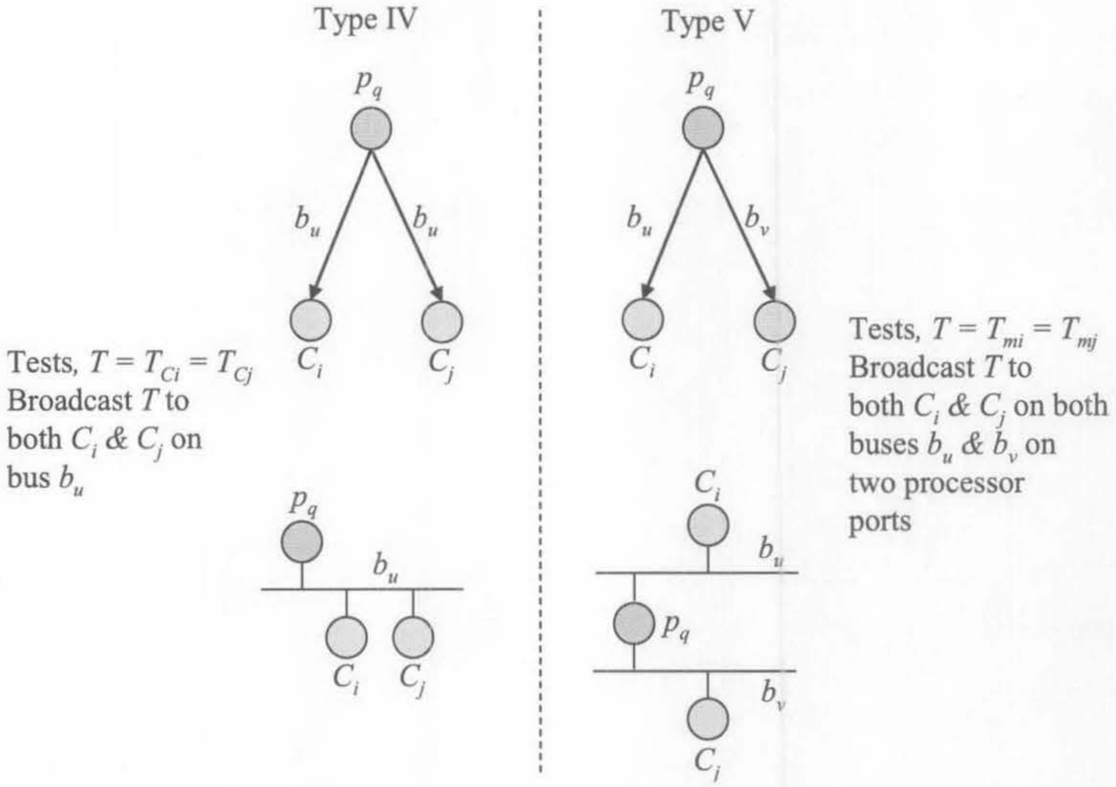


Figure 3.6: Unicast-based test configurations graphs.

3.5.3 Equivalent Test Configuration Graphs

Equivalent test configuration graphs are those that have the same number of bus transactions (data delivery costs) in order to deliver test data from each of the different test sources (processors). Each bus transaction is assumed to have the same cost. For example, Figure 3.8(a) shows two equivalent broadcast-based test configurations. Test configuration belonging to p_r is identical to that of p_q if b_0 and b_1 in p_q are swapped and C_i and C_j in the same TCG are also swapped. In the case of unicast test configurations as illustrated by Figure 3.8(b), the cost is identical since each test configuration requires two bus accesses, $(b_0 \wedge b_1)$ and $(b_2 \wedge b_1)$ respectively, where b_0 and b_1 are local buses for p_q and p_r respectively.

For the purpose of test scheduling, equivalent TCG's are considered to have the same cost and therefore have equal chance of being selected. In the next sec-



Type IV
p_q
b_u
b_u
C_i
C_j
Tests, $T = T_{Ci} = T_{Cj}$ Broadcast T to both C_i & C_j on bus b_u
p_q
b_u
C_i
C_j
Type V
p_q
b_u
b_v
C_i
C_j
Tests, $T = T_{mi} = T_{mj}$ Broadcast T to both C_i & C_j on both buses b_u & b_v on two processor ports
C_i
b_u
p_q
b_v
C_j

Figure 3.7: Broadcast-based test configurations graphs.

tion, the heuristics for power constrained test scheduling is described, in which the equivalent test configurations are evaluated and selected. The selection criteria are formed with the objective of optimizing the bus utilizations.

3.5.4 Eliminating the Obviously Redundant Test Configurations

For MPSoC's with n embedded processors, there exist n sets of test configurations. So, the first step in the scheduling heuristic, we can eliminate some of the obviously redundant test configurations.

Definition 3.2 A test configuration is said to be *obviously redundant* if one of the following conditions is TRUE regarding the required communication channel.

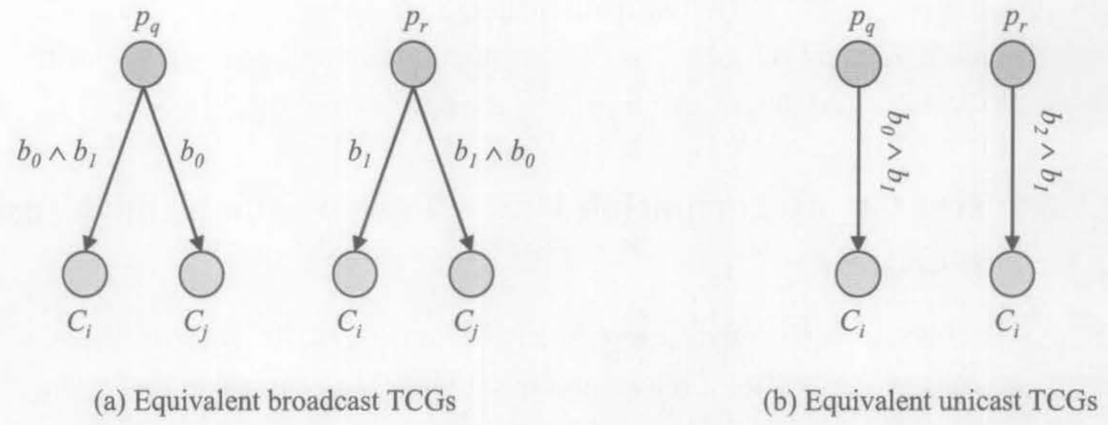


Figure 3.8: Equivalent test configuration graphs.

(i) If there exists a processor cluster, test configurations belonging to all but one of the processors in the processor cluster are said to be obviously redundant.

(ii) The required communication channels of the TCG is a complete superset of another test configuration's required communication channels. ∎

Condition (i) is TRUE under the current assumption that the processor's job is only to delivery test data during the test application. This condition will need to be re-evaluated if the processor is responsible for the real-time generation of the test vectors during the test application (such as the case of processor-based LFSR). Under the current approach, the workload of each processor is assumed to be negligible. Condition (ii) ensures that the local or the closest processor to the CUT, with respect to the number of bus accesses required to deliver the test data, is selected as the test source.

After eliminating the obviously redundant test configurations, the remaining test configurations can be used to classify the CUT's in the MPSoC as either of the following:

1. Cores with unique test configuration.
2. Cores with multiple equivalent test configurations.

For the cores with a unique test configuration, the specified test configuration will be used during the test scheduling process as well as during the test application. For the other cores with multiple remaining equivalent test configurations after the initial elimination, one of the test configurations will be selected during the next process of forming test groups, discussed in Section 3.5.5.

3.5.5 Test Group Formation Under Power-Constrained Test Scheduling

The main objective is to assign a test processor to each of the embedded cores such that the test application time is minimized under a given power constraint, Q_{max}. Since the functional TAM bandwidth is limited, it is important that the data transmission load is also evenly distributed to all the functional buses. Otherwise, one functional bus might be overloaded while other buses remain underutilized; the overloaded bus could unnecessarily delay the test time completion. The test configuration graphs provide an isolated view of the bandwidth requirements on each functional bus for each core. In this section, by utilizing the test configuration graph, the assignment of test processors is performed while simultaneously making sure that the resulting test power never exceeds Q_{max}. We formally define the problem of test processor assignment under power constraint, Ψ_{PP}, as follows:

Problem 3.1 [Ψ_{PP}] Given an MPSoC with p embedded processors, n logic cores, and for each core c, given a TAT, T_c, and a power dissipation, Q_c, at the maximum test frequency, f_{max}, and the set of all test configuration graphs, select the best TCG and test frequency for each core, and assign the core to a test group such that

1. the total test application time is minimized, and
2. the maximum bus utilization for every test group is minimized. ∎

Definition 3.3 A *test group* is defined as a set of cores which are scheduled to be tested concurrently where the start times are identical as shown by "Test Group 1" and "Test Group 2" in Figure 3.9(b). ∎

For hierarchical MPSoC, the test group formation requires a different optimization scheme compared to the flat-bus SoC discussed in Section 2.6.4. With

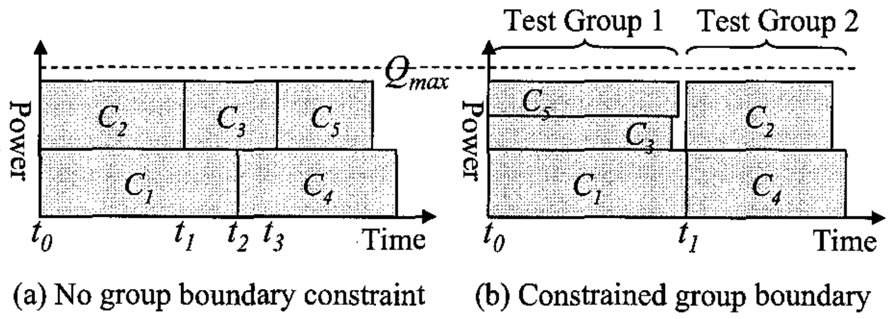


Figure 3.9: Power-constrained test grouping for a bus segment.

hierarchical buses, we can take advantage of the fact that the buses are electrically isolated by the bus bridges. Therefore, test group formation has to take into consideration the congestion in all the bus segments. Otherwise, some bus segments might be congested while other segments remain idle.

Power constrained scheduling for core tests has been performed using various methods, which include bin-packing [19] and preemptive [34] algorithms. The resulting test schedule is typically similar to Figure 3.9(a), where the starting and ending times of a core test are optimally assigned without any constraint. The test application of a new core is started right after another core test is completed, as long as the required resources (i.e. dedicated TAM wires) are available. This unconstrained schedule translates into a complex packet delivery schedule when the functional TAM is considered. This is because we do not consider breaking the functional TAM into independent bus wires which can be dedicated to any specific core. For the example in Figure 3.9(a), five distinct delivery patterns are required. At time t_0, C_1 and C_2 are scheduled in the first packet delivery schedule. At t_1, the test application of C_2 is completed and C_3 is started. This change will be reflected by the packet delivery schedule change. The schedule changes also take place at t_2 and t_3.

The more frequent the schedule changes, the higher the complexity of the overall packet delivery sequences. If the test source is a processor, then the delivery sequences, which include all the timing information for every unique delivery pattern, must be enumerated by the test program. Due to this, in the proposed algorithm the cores are grouped into non-overlapping test groups as shown in Figure 3.9(b), which requires a delivery schedule with only two different delivery patterns, which change at time t_1.

Flexibility in the choice of the test frequency values ($f_c \leq f_{max}$) for a core c could be exploited in order to minimize the impact (on TAT) of the constraint in the test group formation. In order to do this, we revisit the discussion on the relationship between scan power and scan test time, which has been explained in Section 2.6.3. Test power of a core c is directly proportional to the scan frequency. Test time of the same core c, on the other hand, is inversely proportional to the scan frequency. These relationships can be represented by equations (3.1) and (3.2).

$$Q_c \propto f_c \tag{3.1}$$

$$T_c \propto \frac{1}{f_c} \tag{3.2}$$

Therefore, for any non-zero values of f_c, the product $Q_c \times T_c$ always remains constant. Here, the $Q \times T$ product also represents the area of power-time rectangles illustrated in Figure 3.9. Therefore, the rectangle can be reshaped while keeping the area constant. Such reshaping process reflects the change of the test frequency.

For example, in Figure 3.9(b), the test frequencies of C_3 and C_5 are halved, in order to match the TAT of core C_1, so that all C_1, C_3, and C_5 can be scheduled concurrently without exceeding Q_{max}. For a given core, its test power can be reduced (by assigning the test frequency with one of the choices of smaller frequency values) as long as its TAT does not exceed the longest TAT within the test group.

The FormTestGroups procedure in Algorithm 3.1 explains the algorithm for the power constrained scheduling, in which the test groups (Figure 3.9(b)) are formed. The objective is to minimize the test application time, while balancing the bus utilization for all the MPSoC's hierarchical buses within each test group. The procedure begins with forming w_{bus} wrapper scan chains for all cores, where w_{bus} is the bit width of the functional bus. The number of wrapper scan chains that matches the bus bit width is preferred in order to simplify the buffer controller. In such configuration, data can be loaded from the buffer to the scan chains directly, without requiring the serial shifting between the stack and the output register (Figure 2.8(a)). Scan chains are formed such as to minimize the maximum scan-in or scan-out depth using the procedure proposed by [21, 66]. The step is followed by elimination of all redundant TCG's. The following is the definition of a "redundant TCG".

Definition 3.4 A test configuration graph TCG_i of a core C_i is said to be a *redundant TCG* if the set of buses that are traversed by TCG_i is a complete superset of the set of buses that are traversed by another TCG of C_i.

For example, in Figure 3.6, the Type II TCG is a redundant TCG (relative to Type I) for C_i because the set of buses traversed by the Type 2 TCG, $\{b_u, b_v\}$, is a complete superset of those traversed by the Type I TCG, $\{b_u\}$. The redundant TCG's are eliminated because their delivery costs (i.e. the total bandwidth utilization) are always larger and could unnecessarily cause inefficiency in the overall bus bandwidth utilization.

Algorithm 3.1 FormTestGroups (Ψ_{PP})

```
1.  For each core $c \in SoC$, form $n_{sc}$ wrapper scan chains,
    s.t., $n_{sc} = w_{bus}$ and eliminate all redundant TCG's;
2.  Start a new test group $G = \emptyset$;
3.  Set the total power for $G$, $Q_{G,total} = 0$;
4.  Among unscheduled cores, select a core $c$ with maximum TAT and not yet
    attempted for scheduling in the group $G$;
5.  If $c$ (and broadcast pairs, if any) has multiple TCG's {
6.      Select the TCG which results in
        $min\{max\{bus\ utilization\ of\ each\ bus\ in\ the\ TCG\}\};\}$
7.  If (($Q_{G,total} < Q_{max}$ after scheduling $c$) AND
    (at least one of the TCG's buses has min. current utilization)) {
8.      If $G = \emptyset$ {
9.          assign the scan frequency of core $c$, $f_c = f_{max}$;
10.         assign group test time, $T_G = T_c$;
11.       update $G = \{c\};\}$
12.     Else {
13.       reassign maximum $f_c \leq f_{max}$ such that $T_c \leq T_G$;
14.       update $G = G + \{c\};$ }
15.     update $Q_{G,total}$ and the group's current bus utilization;
16.     go to line 4;}
17.   Else {
18.     If (all cores have been attempted in the current group)
19.       Go to line 2;
20.     Else
21.       go to line 4;}
```

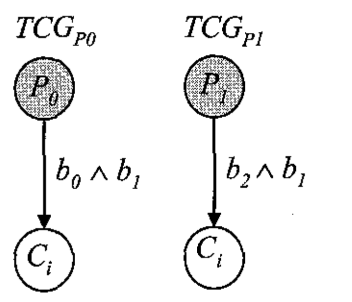
When forming the test groups, the core with the largest TAT among the unscheduled cores is first scheduled to the group and assigned the maximum scan frequency, f_{max}. Using the assumption given in Section 3.3, a core's test frequency, f_m, can only be reduced from the maximum test frequency, f_{max}, such that $f_m \leq f_{max}$. Therefore, the core test time can only be lengthened, resulting in reduced test power. After scheduling the core with the longest TAT first, the TAT (at f_{max}) of the remaining unscheduled cores should be less than or equal to the first core already scheduled in the test group. Since the boundary (on the TAT axis) between adjacent test groups are defined by the first core with the longest TAT (i.e. the *group's TAT*), the other cores to be scheduled into the test group must match as close as possible (without exceeding) the group's TAT. This process ensures an efficient use of the power dissipation. Therefore, when scheduling the subsequent cores into the group, the core scan frequencies are reassigned (line 14) to the highest frequency less than f_{max} such that the TAT of the new group member is less than the group's TAT. The choices of frequency values are constrained by the maximum scan frequency, f_{max}, and the resolution of the frequency divider circuit.

Before deciding whether a core C_i can be assigned to the current test group, line 5 of the FormTestGroups algorithm is evaluated in order to determine the optimum TCG for C_i. With a unique TCG chosen, C_i is scheduled in the current group if it fulfills the following two conditions (line 7). First, the total power dissipation after scheduling C_i (i.e. with the reassigned scan frequency) should not exceed Q_{max}. For the second condition, based on the current required bus bandwidth by the cores already scheduled in the test group, there exists one or more of MPSoC's hierarchical buses, $\{b_{min}\}$, whose bandwidth utilizations are zero or minimum. Core C_i is scheduled in the current test group if at least one of these buses in $\{b_{min}\}$, is traversed by the selected TCG of core C_i. The current bus utilizations are updated, and the same procedures are repeated for the next unscheduled core with the longest TAT.

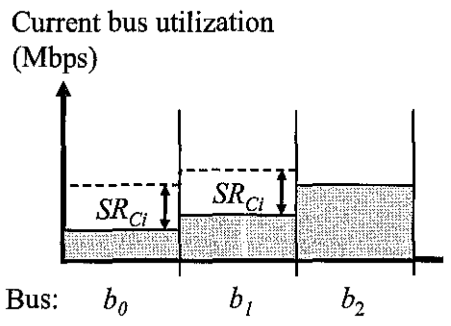
If a core C_i is not scheduled to the current group, it will remain out of the candidate list until either a new test group is created or another core C_j is scheduled to the same test group in which C_i was unable to be scheduled. This is done because of two reasons. First, because the state of current bus utilization changes after C_j is scheduled. Second, because it is better to schedule cores with longer test application times together in the same test group; this is achieved by attempting to schedule the failed cores again once the schedule state changes. This is to preclude the occurrence of large reductions in the test frequency for some smaller cores in order to match the long TAT of the largest core in the current test group.

For the case of cores having multiple TCG's (line 5 of FormTestGroups), for each TCG_i the maximum bus utilization among all the buses after adding the bandwidth requirements of all buses traversed by TCG_i is determined. The selected TCG is one which minimizes the maximum bus utilization. *Bus utilization* of a core refers to the total data rate required on a bus to transport the test data from a test source to the core so that the test data is valid at the core's scan input at every scan clock cycle. The bus utilization of a core c is equal to its scan rate, $SR_c = n_{sc} \times f_c$ in bits-per-second, where n_{sc} and f_c are the number of wrapper's scan chains and the scan frequency, respectively.

Figure. 3.10 illustrates this TCG selection step for a core C_i which has two TCG's, TCG_{P0} and TCG_{P1}. The shaded area in Figure 3.10(b) indicates the current bus utilization due to other cores already assigned to the current test group. By selecting TCG_{P0} for core C_i (Figure 3.10(a)), the resulting bus utilization is more balanced (illustrated by the dotted lines in Figure 3.10(b)) compared to the



(a) Two candidate TCG's for C_i.



(b) Current bus utilization (shaded) and after selecting TCG_{P0} for C_i (dotted lines).

Figure 3.10: TCG selection procedure for a optimum load balancing on the functional TAM.

case if TCG_{P1} were selected, which results in a high utilization in the bus b_2 and a low utilization in the bus b_0. This prevents some particular buses from being overloaded while leaving others underused. The overloaded buses could potentially become a communication bottleneck. This is not an effective load distribution as long as there exists another bus that could share the communication load. This step of dynamically selecting the best TCG for each core during the scheduling process, based on the current bus utilization by other cores, spreads the communication load to all MPSoC's buses. As a result, the maximum utilization of each bus is minimized—the second objective of Problem 3.1 (Ψ_{PP}). After a unique TCG if selected, line 7 is evaluated in order to determine if the core qualifies to be scheduled in the current group. This process of selecting the best TCG is repeated the next time this core is considered for scheduling, if the core is not scheduled in the current iteration.

For the MPSoC cores which share the same test set (called *broadcast pairs*), the test data can be broadcast on the bus in order to further reduce the test application time. The broadcast delivery can be represented by the Type IV and Type V TCG's (Figure 3.7). In this case, the broadcast pairs are treated

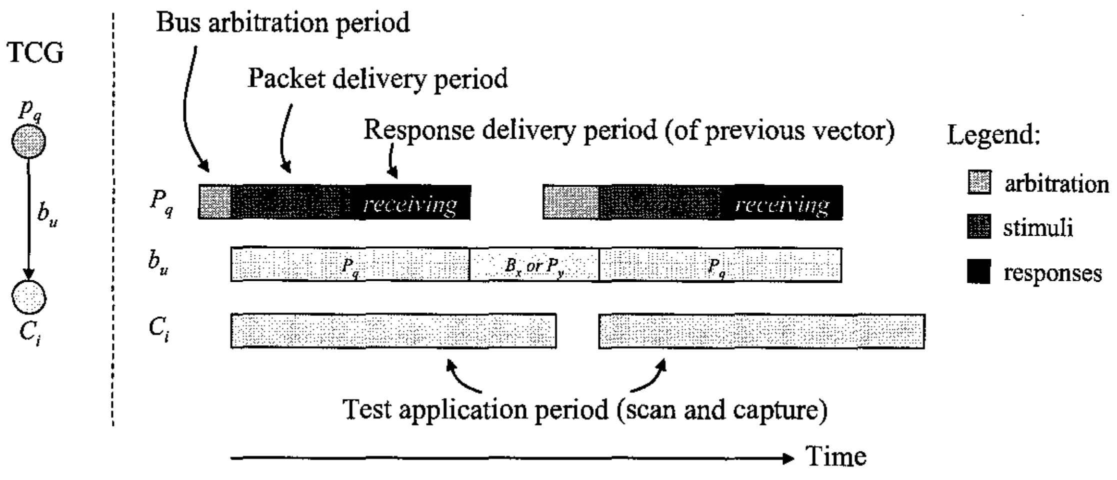


Figure 3.11: Packet delivery timing (local bus transfer).

as a single entity during scheduling. Therefore, when evaluating lines 5-15 of FormTestGroups, the broadcast pairs must be scheduled together in the same test group in order to take advantage of the broadcast capability to reduce the test application time.

Once all cores have been assigned to a test group, each test group goes through the packet set scheduling steps in order to determine the packet delivery schedule which minimizes the test application time of the test group as well as the whole MPSoC.

3.5.6 Complexity of Packet Set Scheduling for MPSoC

Figure 3.11 illustrates the activities on the processor (p_q), bus (b_u) and CUT (c_i) when a test packet is delivered under a Type I TCG. Every packet delivery begins with an arbitration cycle. Upon owning the bus, the processor starts sending the test packet containing a number of bits of the test vectors. During this period, the processor and the bus are occupied. Upon receiving the first bit of the new test vector, CUT starts to scan in the new test vector into the scan chains.

Since the CUT is local to the processor, we will assume that a write followed by a read (write-read) transfer sequence for every test packet delivery is used.

Under the write-read operation, the processor (or bus master) starts by putting the write address on the address bus, followed by the data on the data bus and a write-enable control signal. If the packet size is larger than a single data cycle, the processor utilizes a burst write transfer, where several data cycles can be used continuously. Upon finishing the write transfer, a read cycle follows when the read-enable signal is asserted. This bus delivery sequence is under the assumption that the functional operation supports such a write-read operation.

During the read operation (receiving the test response), processors and bus remain occupied. The CUT continues scanning in the new test vector which is currently stored in the test buffer. This is under the assumption that scan in operation is slower than bus delivery rate. Note that no arbitration and address cycle is required since we can assume that the same address is used for both test data buffer (write mode) and test response buffer (read mode). The bus continues to be owned by the processor using only a single arbitration.

In Figure 3.12, the timing diagram and the activities on the processor (p_q), the buses (b_u and b_v), the CUT (c_i), the bridge ports (B_{pu} and B_{pv}), as well as the core's buffer interface (B_{ci}), are shown for a Type II TCG. This is the case of a non-transparent bridge performing a store-and-forward operation during the delivery of a test packet from the processor p_q to the CUT c_i since the processor is not local to CUT. Note that the processor is only active during the delivery period of the test vector and after a period of inactivity, receiving the test response back from the bridge. During this period (assuming split transfer operation), the processor is free to perform other tasks. Assuming that the two-port bridge can operate both ports simultaneously, we can see that one of the bridge ports remain idle at any given time.

Figure 3.13 illustrates the simplified representation of the packet delivery timing showing only the bus activities for various TCG configurations. Figure 3.13(a)-(d) illustrate the resulting bus activities when delivering a test vector packet and retrieving a response packet for different test configurations as explained below:

1. Transfer to a local CUT
2. Transfer through an intermediate bridge to a CUT
3. Unbalanced broadcast transfer to multiple CUT's

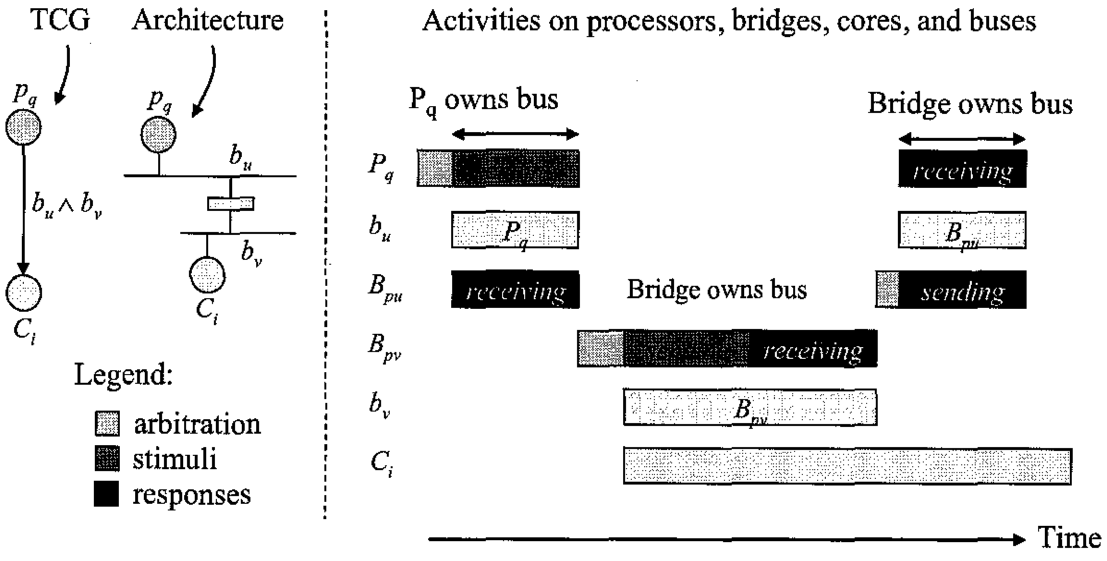


Figure 3.12: Packet delivery timing (through a non-transparent two-port bridge).

4. Transfer through multiple (different) bridges

The packet delivery sequence illustrated by Figure 2.14 is optimized for single-processor SoC's, where packet delivery only utilizes a single processor bus. In those cases, each scheduled packet delivery (and response retrieval) is completed before the next packet in the schedule is delivered. Therefore, the bus is fully utilized from the time packet delivery begins until the time the corresponding response packet is completely received. However, under the hierarchical-bus MPSoC scheme, this is true only for the test configurations which utilize a single local bus as in Figure 3.13(a). As illustrated by Figure 3.13(b), (c), and (d) for packet delivery through hierarchical buses, the packet delivery and response retrieval for a single test packet cannot be completed without forming idle slots on the processor local bus as well as all other intermediate buses.

These idle slots exist between the time a vector packet delivery is completed and the time the corresponding response packet retrieval begins on the same bus. For example, the bus b_0 in Figure 3.13(b) and (c) and both buses b_0 and b_1 in Figure 3.13(d). In other words, the idle slots represent the time duration during which the bus is unused by this test packet. As a result, the bus hierarchy causes

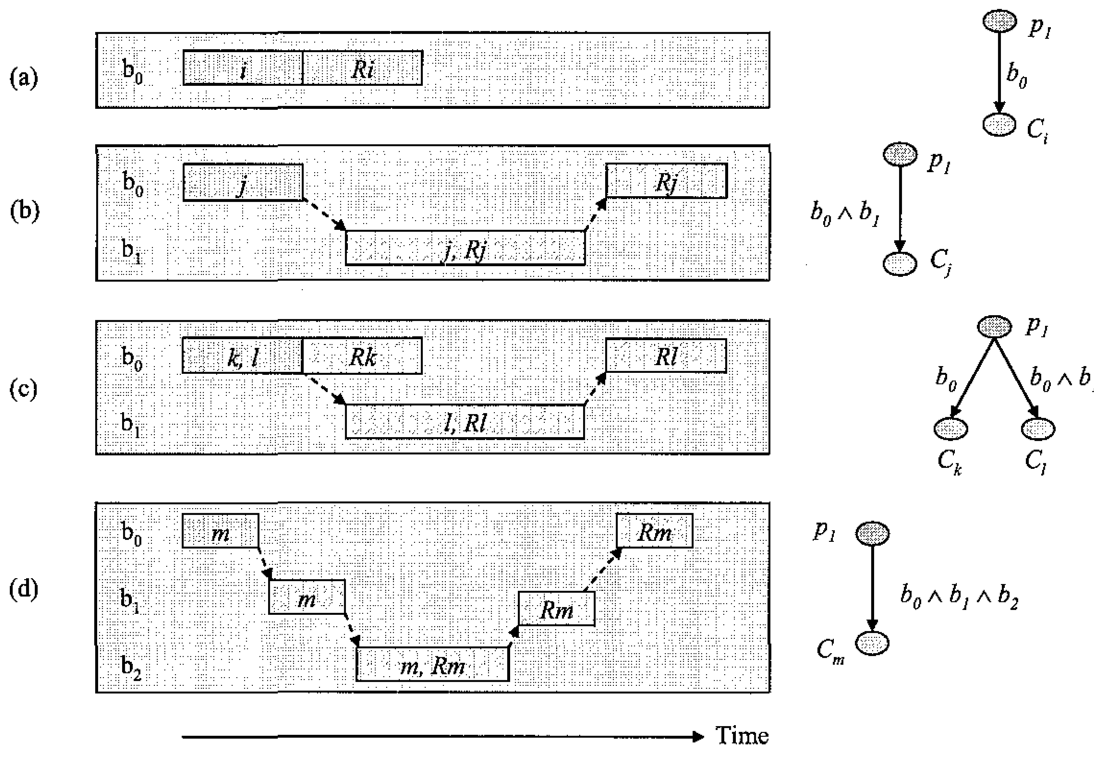


Figure 3.13: Packet delivery timing (bus activities).

a longer round trip delay for every packet. Consequently, the optimized delivery sequence for single bus SoC's (Figure 2.14) is not optimum for hierarchical-bus SoC's. Section 3.6 starts off with this observation to further optimize the test schedule for a hierarchical bus MPSoC.

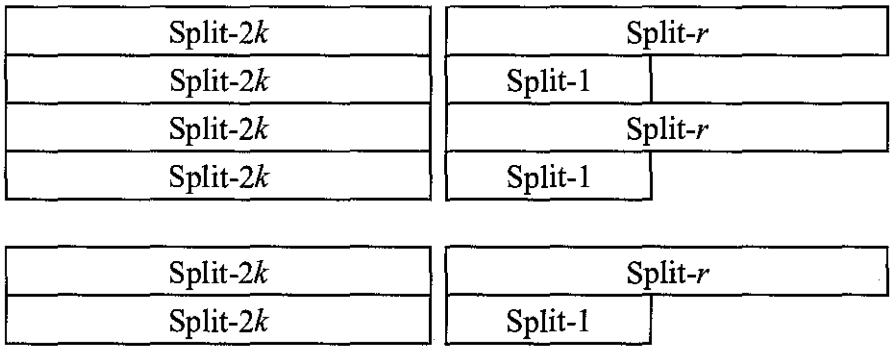
3.6 Optimizing the PASS Algorithm for Hierarchical Bus MPSoC's (MPPASS)

The delivery sequence generated by the PASS algorithm in Figure 2.14 is revisited in Figure 3.14(a). Each $p^c_{i,j}$ represents a time slot on the bus allocated for the transportation of a stimuli and a response packet, where

$c =$ core number (1 to n_i) from split-i group,

$$\begin{array}{llllllll}
p^1_{2k,1} & p^2_{2k,1} & \cdots & p^q_{2k,1} & p^1_{r,1} & p^{1+k/r}_{r,1} & \cdots & p^{1+(d-1)k/r}_{r,1} \\
p^1_{2k,2} & p^2_{2k,2} & \cdots & p^q_{2k,2} & p^1_{1,1} & & & \\
p^1_{2k,3} & p^2_{2k,3} & \cdots & p^q_{2k,3} & p^2_{r,1} & p^{2+k/r}_{r,1} & \cdots & p^{2+(d-1)k/r}_{r,1} \\
p^1_{2k,4} & p^2_{2k,4} & \cdots & p^q_{2k,4} & p^2_{1,1} & & & \\
\vdots & \vdots & & \vdots & \vdots & & & \\
p^1_{2k,2k-1} & p^2_{2k,2k-1} & \cdots & p^q_{2k,2k-1} & p^{k/r}_{r,r} & p^{k/r+k/r}_{r,r} & \cdots & p^{k/r+(d-1)k/r}_{r,r} \\
p^1_{2k,2k} & p^2_{2k,2k} & \cdots & p^q_{2k,2k} & p^k_{1,1} & & &
\end{array}$$

(a) Optimized packet delivery sequence for a non-hierarchical bus. This delivery sequence is also known as a PASS schedule.



(b) The delivery sequence in (a) according to the split group.

Figure 3.14: Packet delivery sequence produced by the PASS algorithm.

$$\begin{aligned}
&\text{where } \textstyle\sum n_i = n \\
i &= \text{split group to which core } c \text{ belongs} \\
j &= \text{packet sequence number for core } c, \\
&\text{where } j \leq i
\end{aligned}$$

The sequence has been optimized for a flat-bus architecture, without considering the delay in bus arbitration and contention with other activities on the bus. In addition, the algorithm assumes that all the activities on the bus are originat-

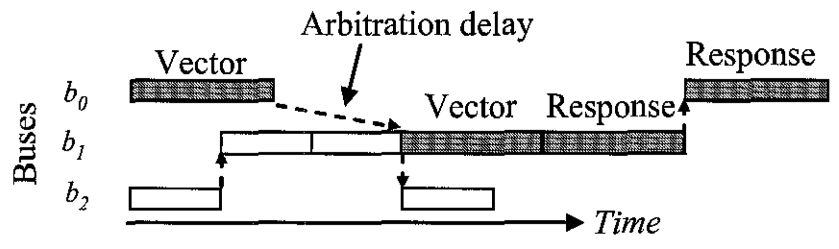


Figure 3.15: Bus contention in a hierarchical bus MPSoC resulting in a delayed data transfer.

ing from a single test processor, thus precluding any bus contention; the timing of vector and response packets is always predictable. For a hierarchical bus MPSoC, the delivery of some packets may not be completed within the allocated time slot indicated by each $p^c_{i,j}$ in Figure 3.14(a). Instead, due to the bus contention and the store-and-forward operation on each packet by the bridges, the completion time of a packet delivery cannot be independently calculated without considering the activities of other processors (and bridges) on the hierarchical buses. Figure 3.15 shows a possible contention scenario on bus b_1 between data packets originating from processors on buses b_0 and b_2. As a result, the data packet coming from bus b_0 is delayed, resulting in a delayed response packet returned to the test processor.

3.6.1 Motivation for Incorporating the Hierarchy Information in the Packet Delivery Sequence

In Figure 3.16, d_i and r_i represent the delivery of data packets and response packets, respectively, belonging to core c_i. Bus b_1 and bus b_2 are the first level and second level buses respectively. Core c_1, which is connected to bus b_2, requires both bus b_1 and bus b_2. Cores c_2 and c_3 require only bus b_1 for test data delivery because they are connected directly to bus b_1. This scenario is under the assumption that the common test source (the embedded processor) for cores c_1, c_2 and c_3 is connected to bus b_1.

In Figure 3.16(a), the delivery sequence is $d_1d_2d_1d_3$, following the schedule

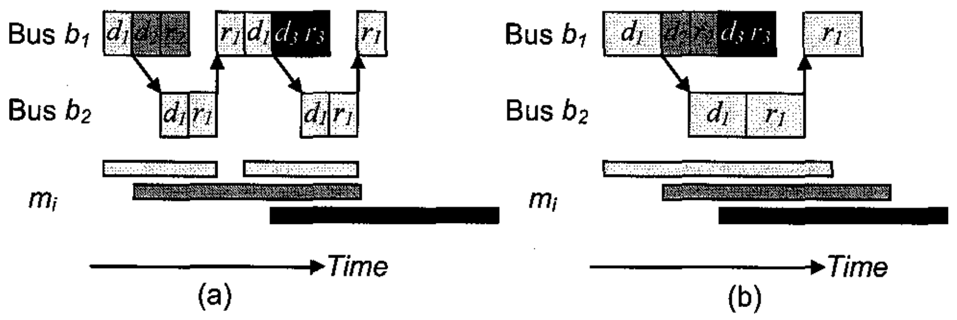


Figure 3.16: Increasing the efficiency of bus utilization.

in Figure 2.14, which is optimized for flat bus SoC's. When determining the delivery sequence, one of the optimization objectives was to minimize the total buffer utilization by increasing the frequency of data delivery for cores which have more test data to be delivered. As a result, two packets for c_1 is scheduled, while only one packet each for cores c_2 and c_3. Under this schedule, packet d_1 needs to be forwarded to the second-level bus through a bridge twice—once for every packet delivered. Each time, the same amount of arbitration time is assumed.

In Figure 3.16(b), after incorporating the bus hierarchy information, the delivery sequence is changed to $d_1d_2d_3$. During the optimization, the split ratio for a core is averaged by the number of buses required for the delivery of a packet belonging to the core. As a result, the split ratio for core c_1 is halved because it requires two bus transfers, while the split ratios of other cores with a single bus transfer remain unchanged. Consequently, the size, the delivery time, and the scan in time of each packet of c_1 are twice those of c_2 and c_3. However, reducing the number of packets reduces the number of bus arbitrations required for the packet delivery.

Under a heavy load on bus b_2, the arbitration time could potentially be the bottleneck on limiting the efficiency of packet delivery schedule by the processor located on the bus b_1. Therefore, reducing the frequency of packet delivery to cores which are far from the processor (in number of bus delivery) reduces the effect of the round-trip delay. In the methodology that we have proposed, the frequency of packet delivery is reduced linearly to the number of different buses that the packet needs to travel in order to get from the processor to the core under test. This is achieved by dividing the split ratio (which was calculated and optimized for flat-bus architecture, explained in Section 2.6.6) by the number of buses in the selected test configuration of each core.

3.6.2 Improving PASS Schedule through Random Permutation

The PASS delivery sequence of Figure 3.14(a) only requires that, for each time slot $p_{i,j}^{c}$, the packet must belong to a core $c \in \{\text{split-}i \text{ group}\}$, shown in Figure 3.14(b). For the case of split groups consisting of multiple cores $c_1, c_2, \ldots, c_{n_i}$, the order (or method) of assigning the delivery slots to each core within the split group is not constrained by the algorithm. For example, the delivery sequence of *B-A-B-C* (Figure 2.11(b)) can as well be changed to *B-C-B-A* without affecting the scan in operation at each core. Since the delivery sequence by each test processor is repetitive, some sequences are better for the hierarchical buses at minimizing the TAT than others.

In general, the number of unique delivery sequences for any test group consisting of n_x, n_y, and n_z modules in the split-x, split-y, and split-z groups respectively is $n_x! \times n_y! \times n_z!$. In other words, there are $n_x! \times n_y! \times n_z!$ ways of forming the packet set schedule (PASS). Each of the unique delivery schedule could result in a different test application time. This is because the contentions between the packet forwarding activities between all modules within the same subgroup as well as between subgroups could delay the delivery of some packets.

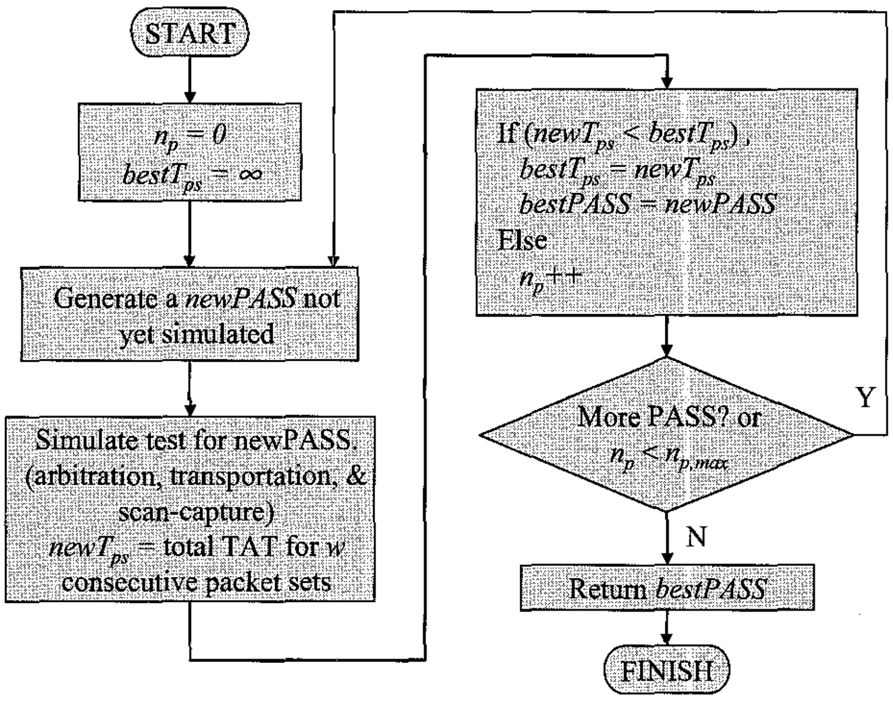
The number of possible delivery sequences grow at a rate greater than exponential, $exp(n)$, where n is the number of cores in a split group. Nevertheless, typical values of n are bounded by several factors. For a MPSoC containing N_c cores under test and N_p embedded processors, the average number of cores in a split group is $N_c/(N_p \times N_{TG} \times N_{SG})$. N_{TG} and N_{SG} are the number of test groups and the number of split groups respectively. In the proposed methodology, $N_{SG} = 3$. N_{TG} depends on the maximum power dissipation. Lower power dissipation means that smaller number of cores forming a test group. For most circuits, the number of possible test configurations is significantly bounded.

Even so, in the worst case, the computation time could still be large. In order to reduce the computation time during the optimization of the packet delivery sequence (or packet set schedule, PASS) a heuristic is explained here. Figure 3.17 shows the flowchart of the packet set optimization methodology. The optimization starts with choosing one of all the possible packet set schedule (PASS) based on the delivery ordering specified in Figure 2.14. Next, the test application based on the current PASS is simulated (refer Section 3.6.3). The resulting test application time is recorded as $newT_{ps}$. For every iteration, if the new $newT_{ps}$ is smaller than the current smallest $bestT_{ps}$, the smallest test time is updated and the corresponding PASS is stored. This process is repeated until one of the stopping conditions described below become true. In every iteration, a new non-repeating (or not-yet-simulated) PASS is randomly generated. After one of the stopping conditions becomes true, the best PASS is returned. The best PASS is finally used as the test configuration for the complete simulation of the test application for the current circuit under test.
Stopping Conditions:

1. All possible permutations of PASS have been evaluated
2. The total number of PASS simulated since the previous PASS which produces a smaller test application time is greater than the maximum preset value, $n_{p,max}$.

In order to minimize the simulation time, each unique PASS is simulated for a small number, w, of packet sets (i.e. w repetitions of *B*-*A*-*B*-*C*, instead of a complete test application for all the test data of cores A, B and C). The value of w is chosen by experiment under the assumption that the interaction between the data packets is cyclic after several repetitions of the packet set delivery from each test processor. Furthermore, if $n_{p,max}$ consecutive PASSes have been attempted without getting any improvement in the TAT, the simulation is stopped. The current best PASS is returned as the delivery sequence for the test group. The $n_{p,max}$ limit is imposed in order to avoid simulating all permutations of delivery patterns, which could be considerably large for test groups consisting of large number of cores.

The flowchart in Figure 3.17 shows the process of determining the best PASS (e.g. *B*-*A*-*B*-*C* or *B*-*C*-*B*-*A*) for each test processor, taking into consideration the



START
$n_p = 0$
$bestT_{ps} = \infty$
Generate a *newPASS* not yet simulated
Simulate test for newPASS. (arbitration, transportation, & scan-capture)
$newT_{ps}$ = total TAT for w consecutive packet sets
If ($newT_{ps} < bestT_{ps}$), $bestT_{ps} = newT_{ps}$ $bestPASS = newPASS$ Else n_p++
More PASS? or $n_p < n_{p,max}$
Y
N
Return *bestPASS*
FINISH

Figure 3.17: Optimizing the packet delivery sequence (MPPASS).

hierarchical nature of the vector and response data packet delivery. A packet delivery simulation is performed for all the delivery sequences that do not violate the group delivery sequence requirements in Figure 3.14(a)—the *valid PASS*es. The valid PASSes can be formed by randomly permuting the delivery ordering within each split group of each test processor. For every permutation of PASS, the test application is simulated for all test processors simultaneously. The simulation environment and procedure are explained in Section 3.6.3. In every iteration, if the new valid PASS returns a smaller TAT, it is recorded as the current best PASS.

The variable $w > 1$ is used because under the MPSoC delivery timing diagram as illustrated by Figure 3.13, packet delivery sequence from different test configurations overlap with each other due to the bus contention and arbitration. This is because any bus master can arbitrate for the bus ownership. Furthermore, a bus master can hold the bus for a maximum duration of BUS_HOLDING_TIME.

In addition, the simulation is assumed to follow the functional bus arbitration scheme, which is designed and optimized for the functional operation. The *first-come-first-serve* arbitration scheme is used for the simulation result reported in this dissertation.

In addition to the overlapping bus activities between packet deliveries within a packet set, there are also overlaps between packet deliveries between adjacent packet sets. Assuming that the overlapping repeats after w consecutive packet sets, we can fairly analyze the actual test application time of the complete test operation without actually simulating the complete test application for each PASS. If w is much smaller than the actual number of repetitions required for the complete test application, we can efficiently speed up the optimization algorithm without sacrificing the accuracy.

Based on the discussion above, the multiprocessor packet scheduling problem, Ψ_{MPPASS}, is formally defined as follows:

Problem 3.2 (Ψ_{MPPASS}) Given a test group, G, consisting of p subgroups, G_j, where $j = 1, 2, \ldots, p$ and p is the number of selected test processors on a hierarchical bus MPSoC. For each subgroup, there are n_j cores $\{c_1, c_2, \ldots, c_{n_j}\}$ to be tested by the j^{th} test processor. For each core c_i, the optimum test configuration graph TCG_{c_i}, the test frequency, f_{c_i}, and the buffer size are given. Determine the optimum packet delivery sequence for each test processor, utilizing the hierarchical functional TAM of the MPSoC, such that the total test application time for the test group G is minimized. ∎

3.6.3 Simulation Environment and Procedures for the Test Data Transportation in Hierarchical Bus MPSoC's

A Multiprocessor Simulator (*MPSim*) is implemented to simulate the transmission of each vector or response packet through the MPSoC's hierarchical buses, considering autonomous processor actions and multiple data packets being transmitted and in transit. The sequence of packet delivery is constrained by the given packet delivery schedule (i.e. PASS) for each test processor in the MPSoC environment. In order to closely mimic the operational environment of the MPSoC during the test application, an event-driven simulation engine (Figure 3.18) has been implemented under the following constraints:

- The delivery sequence by each processor follows the given PASS repeatedly until all test data for all CUT's have been delivered. A new test packet for core C_i is delivered by the processor only when the test response of the previous packet of C_i is successfully received.
- The MPSoC's functional arbitration scheme is used during the test application. For the experimental results presented in this chapter, the *first-come-first-serve* arbitration scheme is assumed.
- Every event requires a separate bus arbitration; a bus request is immediately generated once the data is available at the bus master. For a bridge, the bus request is generated once a complete packet is received or once a queued packet goes to the head of the queue.

In the *MPSim* flowchart (Figure 3.18), the middle (shaded) blocks are the main simulation engine, which keeps track of the simulation time step and processes the events based on their time stamp. The left and right sections show the steps for handling the processor-initiated events and the bridge-initiated events respectively. For both types of events, the packets are forwarded to either the next bridge (vector or response packets), directly to the core (vector packets), or back to the processor (response packet). The next destination of each packet is determined by the TCG chosen for the packet owner (the core/CUT). In most cases, the completion of an event spawns one or more new events, which are pushed into the events queue.

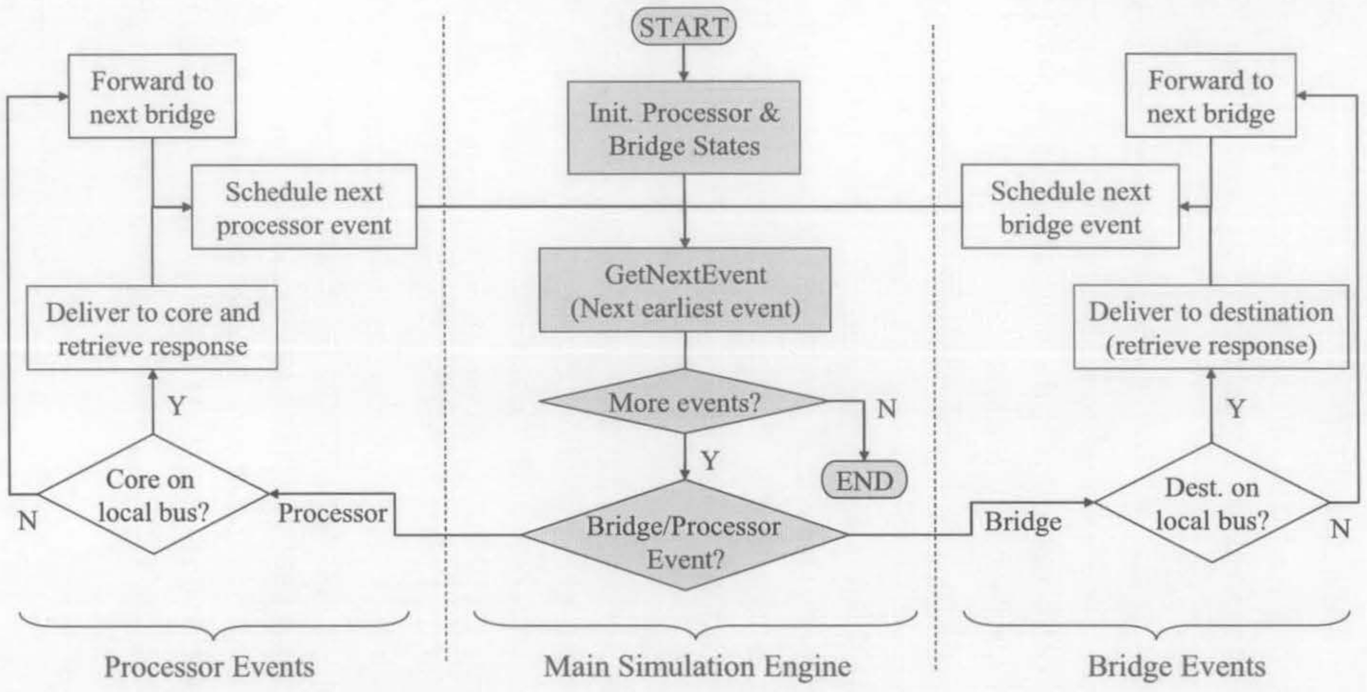

START
Init. Processor & Bridge States
GetNextEvent (Next earliest event)
More events?
N
END
Y
Bridge/Processor Event?
Processor
Bridge
Core on local bus?
Y
N
Deliver to core and retrieve response
Schedule next processor event
Forward to next bridge
Dest. on local bus?
Y
N
Deliver to destination (retrieve response)
Schedule next bridge event
Forward to next bridge
Processor Events
Main Simulation Engine
Bridge Events

Figure 3.18: Event-driven data transfer simulation flowchart of a Multiprocessor Simulator (*MPSim*).

3.7 Experimental Results

In order to evaluate the effectiveness of the proposed methodology, we have conducted experiments on several modified ITC'02 benchmark circuits [1]. The power dissipation information for the selected circuit is obtained from [1, 19][1]. We have additionally added the functional bus information to the selected circuits as follows:

- A single shared bus b_0 is added to connect all the cores, assuming a flat design. A processor core is assumed connected to the bus b_0. This modified circuit is named *NXh1*, where *NX* is the original circuit name. The same test requirements specified in [1] are used.
- For those circuits which have level-2 hierarchy, a single shared bus b_0 is added to connect all the level-1 (hierarchy) cores. The definition for *level* follows the definition in the benchmark suite [1]. Additionally, a local bus b_i is added within each hierarchical level-1 core to connect all the level-2 cores. The bus b_i is interfaced to the bus b_0 through a bridge. This new modified circuit with hierarchical buses is named *NXh2*, where *NX* is the original circuit name. Figure 3.19 shows one of the modified benchmark circuits, *p93791h2*, containing three processor cores, P_0, P_1, and P_2, each connected to separate and isolated bus regions.

In the modified benchmark circuits, we have added a number of processor cores in order to illustrate the ability of the proposed approach to take into consideration the locations of the processors; these processors are not added because of any test requirements but are solely there to illustrate experimentally the superiority and flexibility of the approach we propose. Existing functional processor cores will be used for the actual test application. In the benchmark circuits, the existing cores are not treated as processors in order to provide a fair comparison with previously proposed dedicated TAM based approaches. For the *MPSim* simulator, the values of $w = 5$ and $n_{p,max} = 10$ are used for the results presented in this section.

[1]The unit for power and maximum power, Q_{max}, for is based on an estimate given by [19]. We utilized the same power values in order to offer a comparison with dedicated TAM-based scheduling approaches.

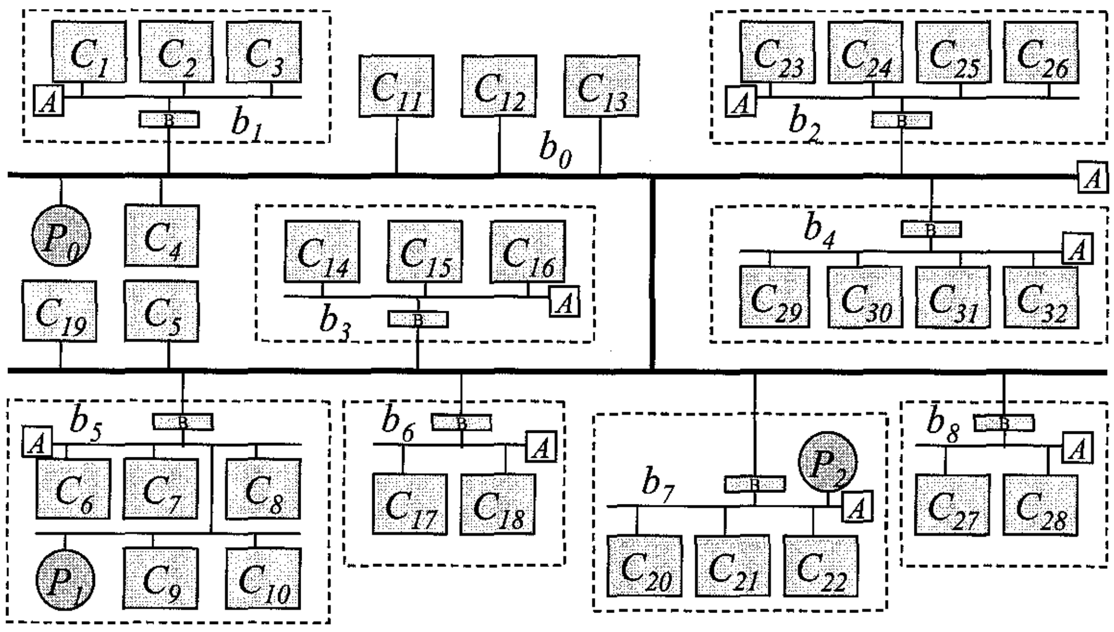


Figure 3.19: Modified benchmark circuit, p93791h2, based on the ITC'02 benchmark circuit [1], p93791.

Among the numerous dedicated TAM-based test scheduling approaches [19, 21, 34, 42, 43, 45, 47–50] available in the literature, the results presented in [19, 37] are used for comparison. Results in [19] are selected because the test scheduling methodology considers both the SoC hierarchy and the test power dissipation as scheduling constraints. Results in [37] ignore the SoC hierarchy, but they are useful when comparing the effectiveness when the functional TAM frequency is larger than the scan frequency; In [37], the virtual TAM frequency is allowed to be as large as twice the scan frequency.

When comparing the test application time with the dedicated TAM-based approach, the processor cores are assumed fault free and tested separately from other embedded cores—the target of testing in this chapter. The TAT's for the processor cores are assumed to be equal for both the dedicated TAM-based and our Integrated Packet Set Scheduling (IPASS) approach, which is based on the functional TAM. The TAT for the processors is therefore not included in the results presented in this section. IPASS dynamically chooses the PASS algorithm for scheduling, when the target system is a single processor SoC with a flat bus architecture. Alternatively, it will use MPPASS algorithm for multiprocessor SoC's with hierarchical buses.

In the following tables, the TAT's are given in milliseconds assuming that the maximum scan frequency, f_{max}, for all cores is 100 MHz. The dedicated TAM-based approaches make use of the maximum scan frequency for all cores. They also disregard the functional buses and rely purely on the added test access architecture; their total TAT's are the same for an SoC with either a flat bus or a hierarchical bus implementation.

Table 3.1 shows the TAT for a single processor SoC with a flat functional bus and bus widths (BW) of 32 and 64 bits. The test applications are simulated for several values of Q_{max}. In the table, "$f_{bus} = f_{scan}$" represents the circuit configurations when the maximum scan frequency (f_{scan}) and the bus frequency (f_{bus}) are both set to the same value of 100 MHz. The columns labeled [19] and [37] show the TAT's for dedicated TAM-based approaches [19] and [37], respectively. Only a single TAT is reported for [37], because the approach does not impose power constraint. It is also much smaller than [19] because it ignores the design hierarchy constraint. The fourth and ninth columns show the results of our functional TAM-based IPASS approach (for 32-bit and 64-bit bus respectively) when the maximum scan frequency and the bus frequency match that of the dedicated TAM approach. The TAT's of our approach and [19] are comparable when the bus frequency is the same as the scan frequency ($f_{bus} = f_{scan}$ columns). They are almost twice those of [37] because the packet transportation on the functional bus cannot overlap; they overlap in the case of dedicated TAM because vector data and response data stream on independent TAM wires.

However, much shorter TAT's are achieved when we allow the bus frequency to be higher than the scan frequency, which cannot be effectively implemented using the dedicated TAM-based approach. To do so, similar buffers must be added, or the TAM frequency can only be increased by also increasing the scan frequency; this would also increase the test power, which may impact the level of test concurrency. Unlike our approach, this hidden cost limits the benefits of increasing the dedicated TAM frequency, as shown by the marginal improvement in the TAT's (columns five and ten, compared to columns three and eight, re-

Table 3.1: The TAT's of the proposed IPASS and dedicated TAM approaches [19, 37], on a flat bus SoC (p93791h1). [37] does not constraint the test power.

p93791h1	$f_{bus} = f_{scan}$			$f_{bus} = 2 * f_{scan}$	
flat-bus	[19]	[37]	IPASS	[32]	IPASS
Q_{max}	BW = 32				
10,000	18.28	9.41	18.44	8.96	9.04
15,000	18.28	9.41	17.34	8.96	8.85
20,000	18.28	9.41	17.35	8.96	8.89
25,000	18.28	9.41	17.63	8.96	9.07
30,000	18.28	9.41	17.78	8.96	9.08
Q_{max}	BW = 64				
10,000	11.17	4.61	8.94	4.53	5.34
15,000	10.15	4.61	8.85	4.53	4.70
20,000	9.58	4.61	8.93	4.53	4.59
25,000	9.65	4.61	9.05	4.53	4.75
30,000	9.45	4.61	9.07	4.53	4.67

spectively). This approach can be efficiently exploited in our functional TAM approach because of the frequency decoupling provided by the buffer-based test architecture. The columns labeled "$f_{bus} = 2 * f_{scan}$" illustrate this advantage; the bus frequency is doubled, while leaving the scan frequency fixed. Since the TAT reduction is constrained by the bus bandwidth limitation, doubling the bus frequency is similar to doubling the dedicated TAM bandwidth, minus the cost of area. The test application time can be reduced, without affecting the test power.

In Table 3.2, *P@b0* represents a the single-processor hierarchical bus circuit where the processor is connected to the level-0 bus, b_0 (Figure 3.19). The TAT's are slightly larger than [19] due to the hierarchy overhead in the delivery time of each test packet. Therefore, the effect of hierarchical buses can be clearly seen when comparing p93791h1 and p93791h2 since in both cases, there is a single test processor on the level-0 bus (b_0). In order to show the benefit of using multiple processors, we analyzed the extreme case scenario, where there is a processor

Table 3.2: IPASS and a dedicated TAM approach [19], on a hierarchical bus MPSoC (p93791h2).

p22810h2	$f_{bus} = f_{scan}$			$f_{bus} = 2 * f_{scan}$	
		IPASS		IPASS	
hierarchy	[19]	*P@b0*	*P@All*	*P@b0*	*P@All*
Q_{max}	BW = 32				
10,000	18.28	26.97	15.51	13.47	7.83
15,000	18.28	20.15	9.51	10.07	4.83
20,000	18.28	20.39	7.37	10.20	4.27
25,000	18.28	18.95	5.31	9.47	3.24
30,000	18.28	18.89	5.31	9.44	3.24
Q_{max}	BW = 64				
10,000	11.17	13.47	7.83	7.13	5.69
15,000	10.15	10.07	4.83	5.05	3.79
20,000	9.58	10.21	4.23	5.11	3.51
25,000	9.65	9.50	3.20	4.72	2.82
30,000	9.45	9.44	3.20	4.78	2.82

in every isolated bus region (*P@All*). The bus hierarchy allows simultaneous delivery of the test data and reduces contention on the bus access. For typical MPSoC configurations, the TAT is expected to be between the two extreme cases (*P@b0* and *P@All*). Simulation results for some randomly assigned number and locations of processors demonstrate this expected trend.

Table 3.3 and Table 3.4 show the TAT's for flat-bus p22810h1 and hierarchical bus p22810h2 SoC's, respectively. Similar trends are observed for this circuit. However, smaller differences are observed between *P@b0* and *P@All* because p22810h2 has only three bus regions, as compared to eight bus regions for p93791h2. Correspondingly small variations are also observed between the flat-bus and hierarchical bus SoC's due to the same reason.

The area cost of the proposed buffer-based test architecture can be estimated in terms of the number of flip-flops for the buffers. For all the circuits in Table 3.1

Table 3.3: Flat bus SoC (p22810h1). No power constraint in [37].

p22810h1	$f_{bus} = f_{scan}$			$f_{bus} = 2 * f_{scan}$	
flat-bus	[19]	[37]	IPASS	[32]	IPASS
Q_{max}	BW = 32				
3,000	4.83	2.23	4.34	2.19	3.06
4,000	4.80	2.23	4.34	2.19	2.93
5,000	4.72	2.23	4.53	2.19	2.74
6,000	4.76	2.23	4.67	2.19	2.46
10,000	4.73	2.23	4.32	2.19	2.21
Q_{max}	BW = 64				
3,000	3.09	1.33	3.06	1.12	2.93
4,000	3.24	1.33	2.94	1.12	2.65
5,000	3.22	1.33	2.74	1.12	2.33
6,000	2.50	1.33	2.49	1.12	1.88
10,000	2.36	1.33	2.20	1.12	1.36

to Table 3.4, the buffer sizes per core (for BW=32), averaged over all Q_{max}, are shown in Table 3.5. The area cost on the controller and the boundary scan cells are comparable to the IEEE 1500 wrapper architecture; therefore, it is not included.

3.8 Conclusion

We have proposed a test scheduling methodology for core-based testing of SoC's based on the utilization of the functional buses as functional TAM's. The proposed method can handle both flat bus single processor architectures and hierarchical bus multiprocessor architectures. The bus hierarchy information is efficiently incorporated into the methodology by representing the resource graph with the *test configuration graphs*—a concept introduced in this chapter.

It was shown that the hierarchical bus architecture introduces additional delay in the delivery of a test packet, which prolongs the overall test application time.

Table 3.4: Hierarchical bus MPSoC (p22810h2).

p22810h2 hierarchy	$f_{bus} = f_{scan}$			$f_{bus} = 2 * f_{scan}$	
	[19]	IPASS		IPASS	
		P@b0	*P@All*	*P@b0*	*P@All*
Q_{max}	BW = 32				
3,000	4.83	4.89	4.23	3.79	3.66
4,000	4.80	4.61	3.06	2.65	2.67
5,000	4.72	4.81	2.88	2.47	1.87
6,000	4.76	4.65	2.96	2.35	1.87
10,000	4.73	4.69	2.70	2.30	1.37
Q_{max}	BW = 64				
3,000	3.09	3.59	3.72	3.54	3.72
4,000	3.24	3.16	2.67	3.05	2.67
5,000	3.22	2.48	1.87	1.93	1.87
6,000	2.50	2.35	1.87	1.68	1.82
10,000	2.36	2.37	1.37	1.30	1.33

Table 3.5: Average input buffer sizes per core.

Circuit	p93791h1	p93791h2	p22810h1	p22810h2
Min.	99.20	89.79	106.06	107.65
Max.	99.39	98.00	112.00	113.15

Subsequently, the use of multiple processors embedded within the bus hierarchy annuls the negative effects of the hierarchical bus architecture on the overall test application time.

Chapter 4

Network-on-Chip Interconnect Architecture

4.1 Introduction

System-on-a-Chip (SoC) architecture has been the *de facto* design style used by most electronic device manufacturers for many years. Ever since Gordon Moore predicted in 1965 that the circuit density would double roughly every eighteen months, chip developers have been relentlessly tracking this trend. The International Technology Roadmap for Semiconductors (ITRS) 2007 [2] provides a technically sound projection of the future of SoC's for consumer products. Figure 4.1 (from ITRS 2007 System Drivers) gives a generic template for the SoC consumer product. The design template consists of a main processor, multiple processing engines (PE), main memory, and peripherals. A PE is a processor customized for a specific function; large-scale and highly complicated functions such as MPEG, encryption/decryption, et cetera are suitable for implementation as a PE.

The projected quantified design complexity trends for the SoC architecture in Figure 4.1 is shown in Figure 4.2, using the following model assumptions.

1. There will be one main processor with approximately constant complexity.
2. Peripherals will also maintain constant complexity.
3. PE complexity remains constant, while the number of PE's continue to grow subject to a die size of 64 mm^2.

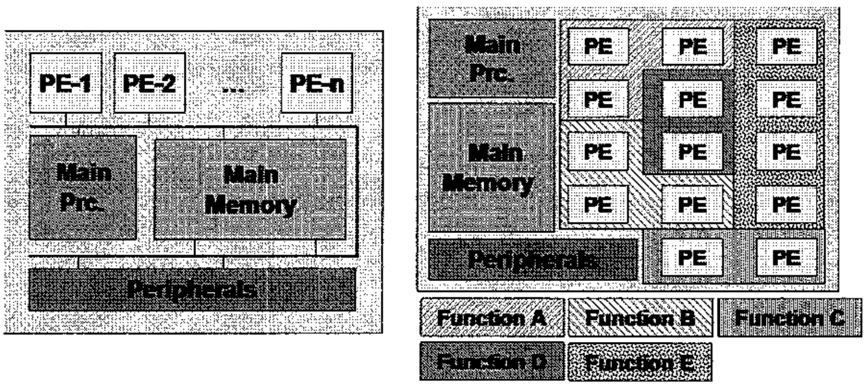


Figure 4.1: ITRS 2007's SoC consumer architecture template.

4. The amount of memory is assumed to increase proportionally with the number of PE's.

A single most obvious conclusion from the ITRS projection is that the number of SoC's IP cores are to grow in a Moore's Law-like exponential growth. What's not shown in the SoC template in Figure 4.1 is the interconnect architecture. It is because the functions of the SoC blocks are (or should be) independent of the interconnect architecture. Obviously, as the number of cores increase, so will the interconnect bandwidth requirement. Interconnect bandwidth is one of the major challenges of multi-core design.

Success of a design relies partly on the use of appropriate design and process technology. In a complex many-core design, the ability to efficiently interconnect all the components becomes a major factor in the design's success. As design complexity has increased, the interconnects have evolved from a single bus to multiple hierarchical buses, and recently to Networks-on-Chip (NoC) [73]. The effectiveness of the Internet Protocol networks inspired the birth of the NoC, since it can provide large on-chip bandwidth for inter-core communications; its modular infrastructure eases the transition effort from the traditional bus-based architecture. In [73], the authors highlighted several pioneer NoC architectures as well as the test-related challenges that must be overcome to promote the adoption of NoC as an SoC interconnect.

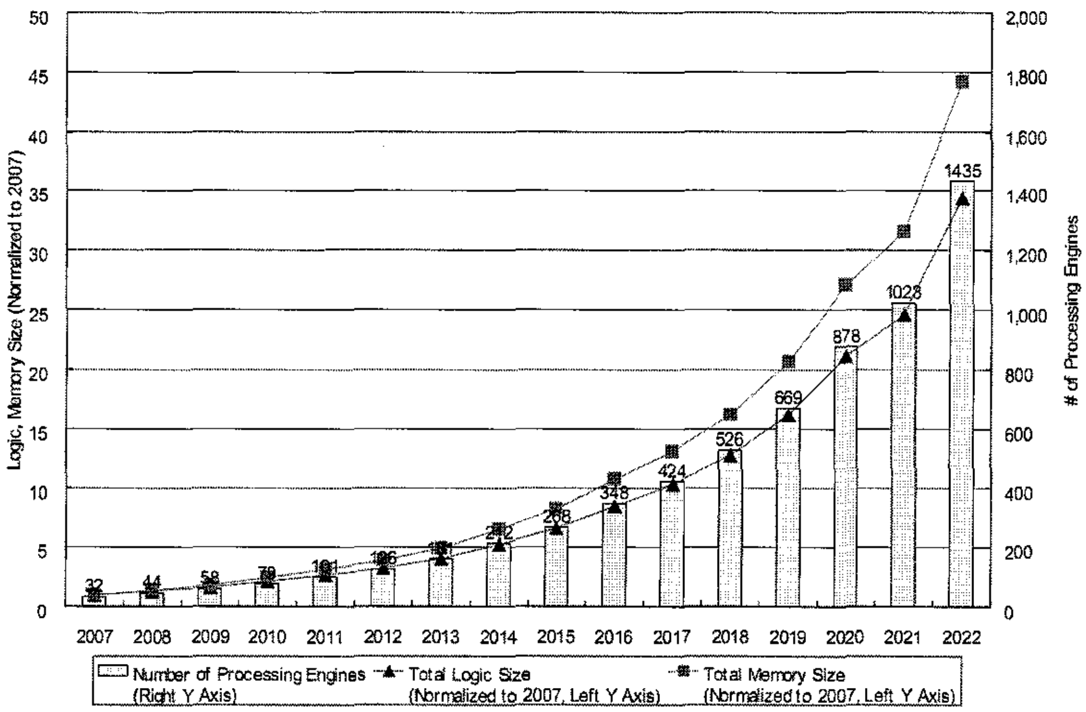


Figure 4.2: ITRS 2007's SoC consumer design complexity trend.

Many NoC architectures have been proposed such as SPIN [75], OCTAGON [76], PROTEO [77], CLICHÉ [78], Æthereal [79, 80], SoCIN [81], SoCBUS [82], xPIPES [83], NOSTRUM [84], QNoC [85], and HERMES [86]; all are based on synchronous communication between nodes. Several other types of NoC's such as CHAIN [87], NEXUS [88], ANoC [89], and MANGO [90] are based on Globally Asynchronous Locally Synchronous (GALS) communication. The copious NoC architectures highlight the growing interest in NoC as a next generation SoC interconnect. Section 4.4 explains briefly some selected NoC architectures.

Before discussing the specific network architecture, we will briefly explain the NoC communication services and the NoC network topologies.

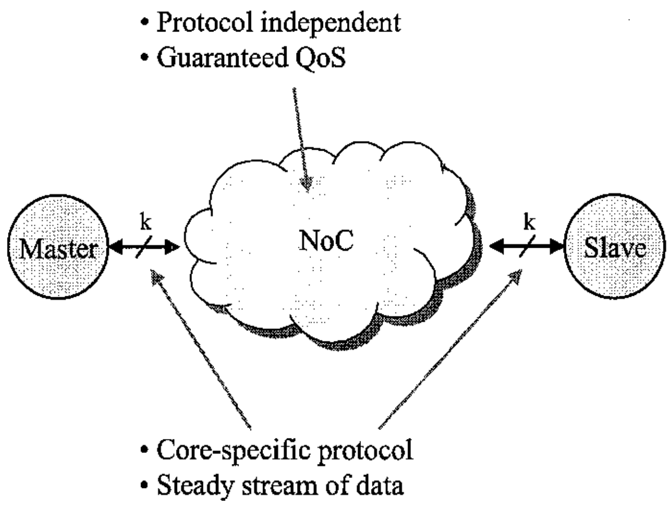


Figure 4.3: The abstract representation of a master-slave data communication model.

4.2 NoC Communication Services

In the traditional bus-based communication infrastructure, data transfer occurs between a master (data source, such as a processor) and a slave (a data sink such as a memory or an I/O controller) when a read or write instruction is executed. When a write transaction is executed, the data is made available on the data bus. This data is instantaneously available at the slave's data bus, ready to be copied. When replacing the bus with the packet-switching-based NoC, the same level of transparency is required. This is illustrated in Figure 4.3. What goes on within the NoC itself must be hidden from the master and the slave. In other words, the master and the slave should not care whether the interface between them is a bus or any other types of interconnects, including the NoC.

Figure 4.4 illustrates one of the services (packetization) provided by the NoC, transparently. Packetization is the process of breaking up large transaction messages into smaller packets, which can be handled more efficiently inside the NoC. The smaller packet sizes is more efficient because of the following.

- Smaller packets require smaller buffers at the routers.

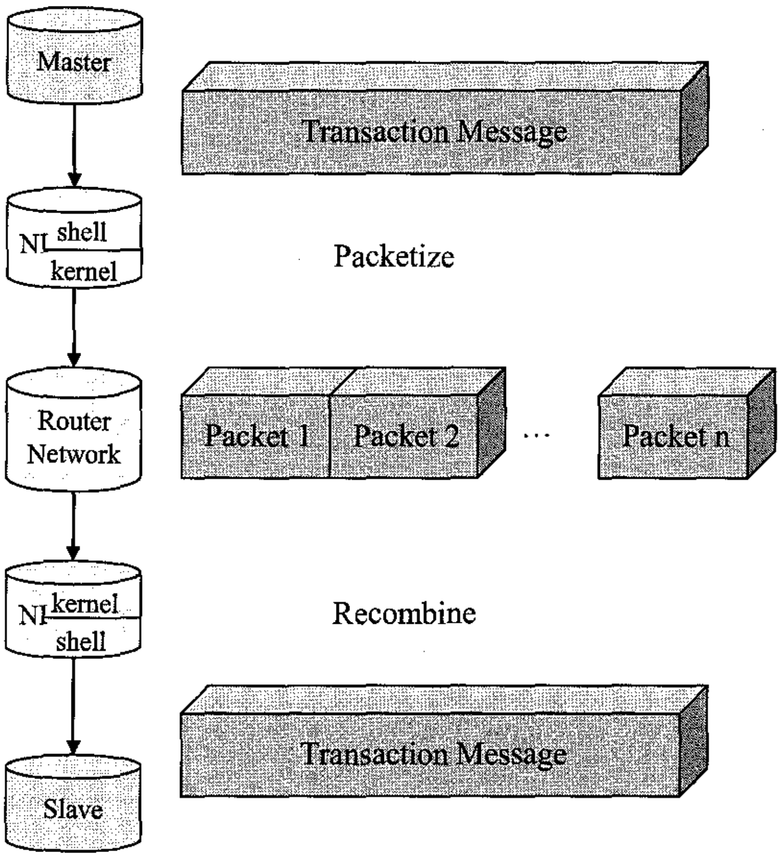
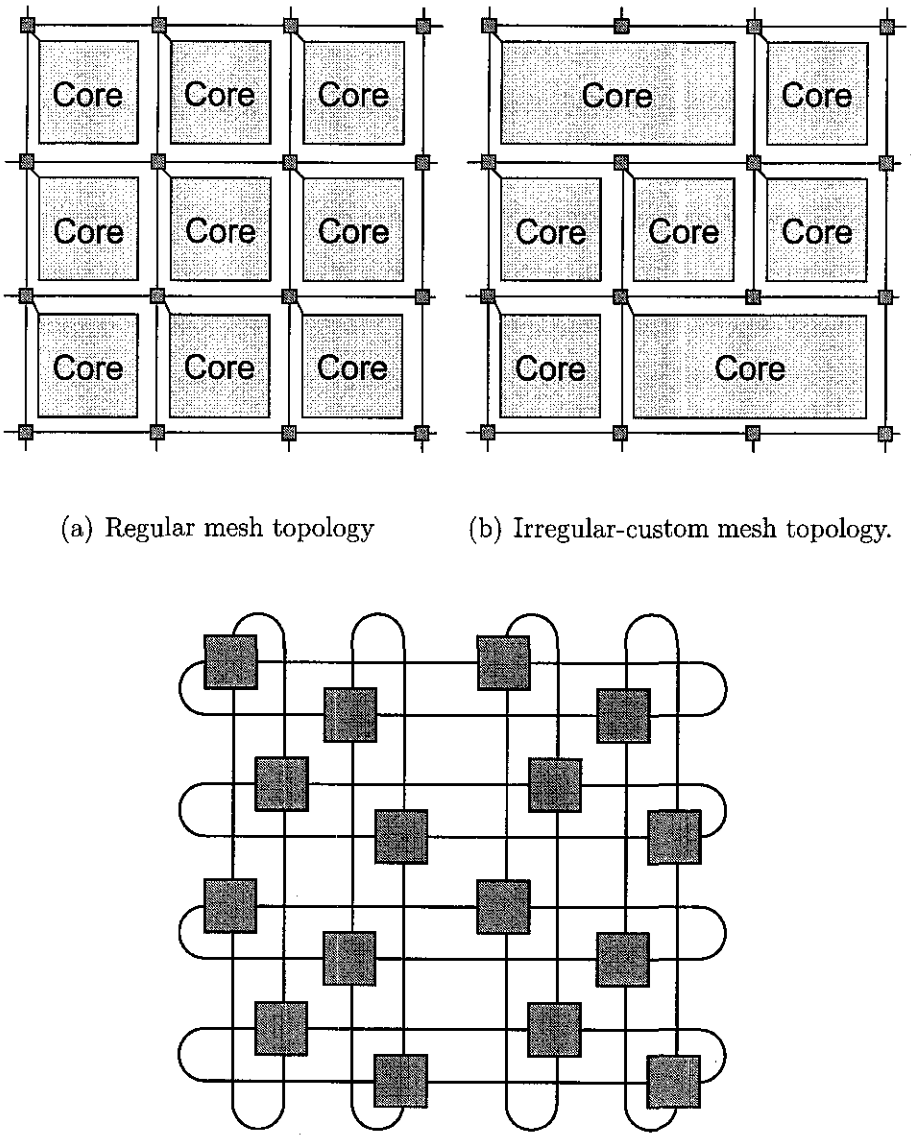


Figure 4.4: Transparent packet-switching services provided by the NoC between a master and a slave.

- Smaller packets occupy less time on the path. Therefore, the resources can be fairly allocated to all that require its resources since it reduces the possibility of resource hogging.
- In case of an erroneous transmission, only a small packet requires retransmission.

There are many other critical and important services provided by the NoC include. Among others, they include

- Error-free transmission, through error checking and retransmission.
- Timing guarantee, such as latency.
- Bandwidth guarantee by means of a virtual connection.



(a) Regular mesh topology (b) Irregular-custom mesh topology.

(c) Folded torus topology

Figure 4.5: Mesh network topology.

4.3 Network Topologies

Like the Internet Protocol network counterpart, NoC network consists of a network of routers with multiple I/O ports. Through these I/O ports, the routers are

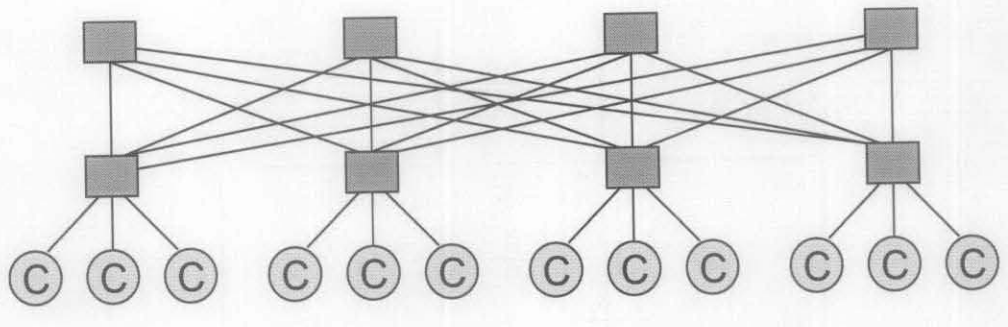


Figure 4.6: Fat-tree topology.

interconnected to each other to forming a topological network map. Some of the most popular topologies are regular mesh, folded torus, irregular mesh-custom topology, and fat-tree. These network topologies are illustrated in Figs. 4.5 and 4.6. Other network topologies are also used such as star network and octagonal-shaped network [76], among others.

Each topology has its own advantages and disadvantages. For example, the folded torus topology requires slightly more routing overhead compared to the regular mesh topology. For a sixteen-router network (4×4), the maximum distance between any two routers in the folded torus network is four hops. In the regular mesh network, the maximum distance for the same network size is six hops. For the two-stage fat-tree network, the maximum distance between any pair of routers is two hops.

These are only a few among many variations of NoC topology being considered and/or used by the different NoC architectures developed by the academic and industrial researchers.

4.3.1 NoC Routing and Forwarding

NoC Networks are composed of routers or switches, which transport data from one place to another. The switches are composed of a switching mechanism and several buffered ports. Figure 4.7 shows a five-port switch architecture of the CLICHÉ NoC [78]. Most other NoC architectures are also using the same router configuration. The four horizontal and vertical facing ports are usually connected

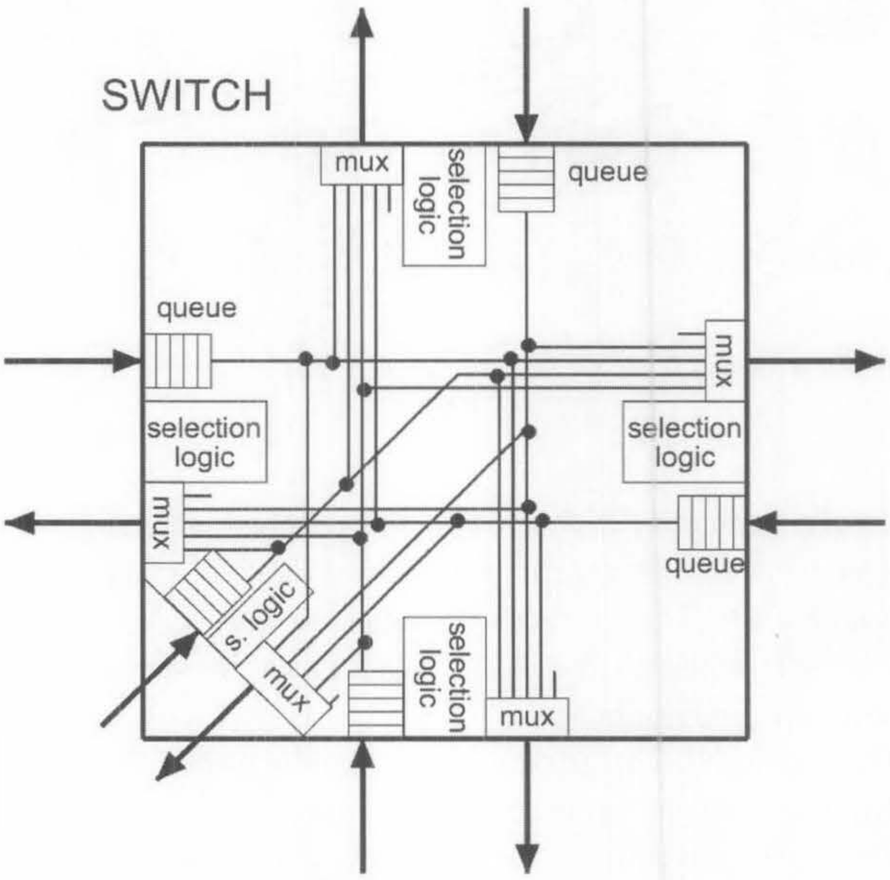


Figure 4.7: Block diagram of a switch architecture for CLICHÉ NoC [78].

to the neighboring switches, while the slanted port connects to an IP core.

Communication between the IP core and other cores in the system goes through one of the ports. At each switch port, the data is buffered and forwarded to the next switch until it reaches the destination according to the routing algorithm defined. Unlike the Internet protocol network that uses adaptive and intelligent routing protocol, the NoC switch (or router) typically implements a fixed routing algorithm. This is because of the following.

1. A fixed routing algorithm requires less hardware resources. Since the router is on-chip, the area occupied by the router is an important factor when deciding the routing algorithm.

2. The NoC network topology is fixed. Therefore it is possible to find an efficient routing algorithm for the rigid network topology. The amount of traffic generated by each node is, however, unknown at design time; nevertheless, a statistical estimation based on the high-level circuit information can be made to help enhance the routing algorithm.

To be precise, we can define *routing* as the task of determining the path (i.e. a series of routers) that a data packet will take from a source node to a destination node. *Switching* is a flow-control technique or the process of transferring the data bits from an input port to the next output port. There are several types of switching methods.

1. *Store-and-forward* switching technique buffers a complete data packet in the intermediate station to be forwarded at a later time to the final destination or to another intermediate station. Later time could be as soon as the whole message is received or until the next channel is available for use. At the intermediate station, or node, the integrity of the message is verified before forwarding it. The weakness of this method is that there is large latency because of the buffering time before forwarding to the next destination. It also requires a considerably large buffer to store all the incoming data.
2. In a *virtual cut-through* switching method, instead of buffering the whole data packet (or frame), the data is forwarded before the whole frame has been received, normally as soon as the destination address is processed. However, when forwarding to the next router, the packet is only forwarded when there is buffer space for the whole packet. Therefore, in the worst case, it requires a buffer size of equivalent to the store-and-forward scheme, or one packet size. The advantage of this technique is that it reduces latency through the switch, therefore higher throughput can be achieved. The disadvantage is that it decreases the reliability because the message integrity is not verified before forwarding. A higher level integrity check such as an end-to-end mechanism will be required.
3. In a *wormhole* switching, a packet is divided into smaller flits (or *fl*ow control un*it*). When a header flit, which contains the information about the next destination, of the packet arrives at the input channel, it is forwarded to an output channel as soon as the output channel is available, and at least

one flit-size buffer is available at the next hop (even if the payload (data) flits are still flowing through the network). Most of the NoC architectures utilize the wormhole switching because of the low latency and small buffer requirements.

4.4 NoC Architectures

This section explains briefly two NoC architectures—SoCIN and Æthereal—that are commonly used by academia and industry researchers for test wrapper design and test scheduling. In Chapter 6, we compare our flexible scheduling algorithm based on the flexibility of the Æthereal architecture and several other research works that are based on the less flexible SoCIN architecture.

4.4.1 SoCIN Network Architecture

A network-on-chip can be described by its topology and by the switching and routing strategies. Other packet-switched related services, such as flow control and arbitration, also define the NoC architecture. The (System-on-Chip Interconnection Network) SoCIN architecture can be built on a 2-dimensional direct topologies such as regular mesh and folded torus (Figure 4.5). In a SoCIN [81] based system, cores communicate by sending and receiving request and responses messages, i.e. read and write transactions. At the core-SoCIN interface, OCP/SoCIN and VCI/SoCIN wrappers are implemented to allow integration with cores that are compliant with these standard SoC interfaces. At the SoCIN side, the wrapper includes a static routing table which is used when constructing the message headers.

The SoCIN specification defined in [81] uses a deterministic and source-based routing. Each packet sender must determine the path to be used by a packet and include the corresponding routing information in the packet header. The SoCIN uses XY-routing in order to minimize complexity. Further, such choice constraints the topology to only two-dimensional grid or mesh. In the XY-routing, a packet going from a source to a destination must first travel in the X direction. When it reaches the destination's column in the mesh, the packet follows the Y direction until it reaches the destination.

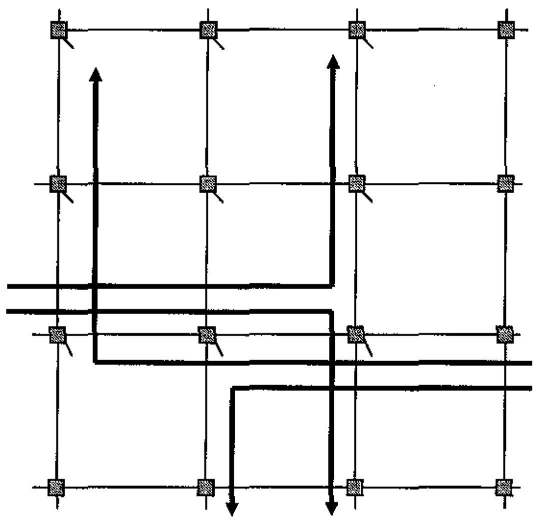


Figure 4.8: Four possible routing directions for XY-routing algorithm. The limited choice of routing paths could easily cause congestion at certain locations while leaving other locations remain unused.

The simple and cheap routing algorithm ensures a deadlock-free routing because no forwarding loop is possible. Figure 4.8 illustrates the XY-routing scheme for a 4×4 network. The figure shows the four possible horizontal direction first (X), then vertical direction (Y) routing paths. For the example shown, some segments of the network are congested. For the given condition, the NoC has no other option but to route the packets through the paths shown. The XY-routing algorithm, while effective in terms of area cost, is inflexible and sometimes inefficient.

SoCIN uses wormhole switching. A message is broken into packets, and packets are broken into flits. Wormhole switching minimizes both buffer requirements and switching latency. In addition, SoCIN uses the handshake flow control protocol and distributed arbitration scheme at each RASoC (Router Architecture for System-on-Chip) network router. Similar to the generic switch architecture in Figure 4.7, RASoC router has five bidirectional ports, with a crossbar switch matrix interconnecting all the bidirectional ports. The routing decision is based on the source-routing path defined in each packet's header by means of a source routing.

A more flexible NoC in terms of network topology and routing algorithm is implemented by the Æthereal NoC, discussed in the next section.

4.4.2 Æthereal NoC Architecture

The Æthereal NoC utilizes a combination of a guaranteed throughput (GT) routers and best effort (BE) routers [79] with wormhole switching, and defines a modular network interface [80]. Such combination of GT and BE routers enables the Æthereal NoC to provide both guaranteed and best-effort services. Guaranteed services are typically used for streaming and critical traffic such as video processing IP, which typically requires lossless, in-order video stream with guaranteed throughput. Best effort services can be used for non-critical traffic.

The GT router guarantees uncorrupted, lossless, and ordered data transfer, and both guaranteed latency and throughput over a finite time interval. The GT router uses a slot table to avoid contention on a link, to divide bandwidth per link between connections, and to switch data to the correct outputs. There is a logical notion of synchronicity; all routers on the network are in the same fixed-duration slot. Therefore, in order to guarantee throughput, the connections (i.e. virtual circuits or virtual channels) are allocated time slots; more allocated time slots means more guaranteed bandwidth.

This thesis assumes that the NoC in consideration is functionally equipped with such bandwidth allocation scheme. The Æthereal NoC employs a time-slot-based time domain multiplexing (TDM) scheme, where a central arbitrator takes charge of the bandwidth allocation for the whole NoC. Figure 4.9 shows the conceptual view of the token-ring-based TDM time slots. Each globally synchronous router port has an identical set of time slots. As virtual channels (VC) are established, sequential slots are reserved on the adjacent routers along the VC path. When connections terminate, slots are freed. The number of slots reserved represents the amount of guaranteed bandwidth reserved. Fig 4.9 shows five VC's, VC_1, VC_2, VC_3, VC_4, and VC_5, with 1 Gbps, 1 Gbps, 2 Gbps, 3 Gbps, and 3 Gbps bandwidths respectively, assuming that the aggregate channel bandwidth is 10 Gbps.

Figure 4.10 shows a System-on-Chip model that implements an Æthereal NoC

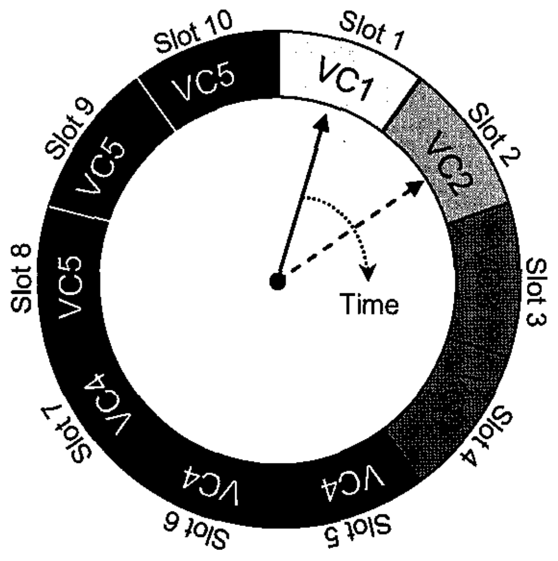


Figure 4.9: Bandwidth sharing is supported by the time slot-based Time Domain Multiplexing (TDM) scheme implemented by Æthereal NoC.

consisting of four routers $R0 - R3$ and network interfaces (NI) [79, 80] as its communication architecture. Among others, the task of the NI is to translate the data format that is passing through. Two of the external ports are labeled *I/O port 1* and *I/O port 2*, which are used in the proposed approach to interface the external ATE ports to the NoC.

Access to the IP cores through the NoC network of routers are possible by means of *shared-memory abstraction*. What this means is that the communication port of the IP cores (connected to the NI) are assigned a unique physical address. The communication port is accessed by means of the standard read and write transactions. The services provided by the NoC are completely transparent. This property makes the transition from bus-based architecture to NoC interconnect architecture seamless. No special changes need to be made to the design.

A transaction-based protocol is implemented in order to provide backward compatibility to existing on-chip communication protocols such as Advanced eXtensible Interface (AXI) [91] and Open Core Protocol (OCP) [92]; this also allow for an efficient implementation of future NoC protocols. The network interface

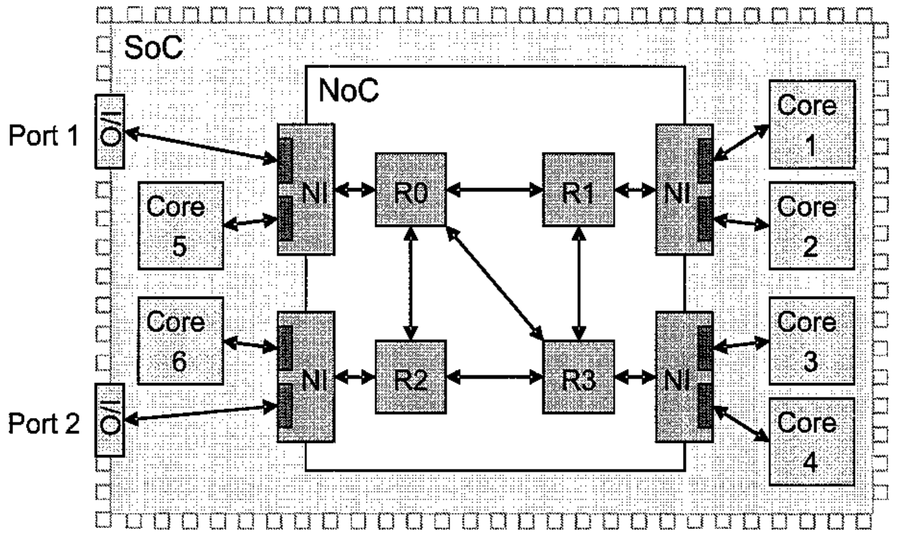


Figure 4.10: SoC model based on the Æthereal NoC.

is split into two components in order to optimize its implementation. The *NI kernel* (NI_k) implements generic communication services such as establishing the communication channel by means of virtual connections, breaking messages into smaller packets and scheduling the packets to the routers, implementing the end-to-end flow control and clock domain crossings. These are the high-level services defined by the OCP communication model.

The lower level services are implemented as a separate entity, the *NI shells* (NI_s). NI shells implement the specific functions of various on-chip protocols, the transaction ordering for connections, and other issues specific to the protocol offered to the IP. The NI supports multiple communication protocols required by the IP cores. Different communication protocols can be implemented for different IP cores as NI shells.

Figure 4.11 shows a simplified timing diagram of an AXI burst write transaction [91]. In order to reuse the NoC during test, the ATE needs to communicate with the CUT using the read/write transactions. The write transaction variables and the IP core model considered in this dissertation are explained in the subsequent chapters.

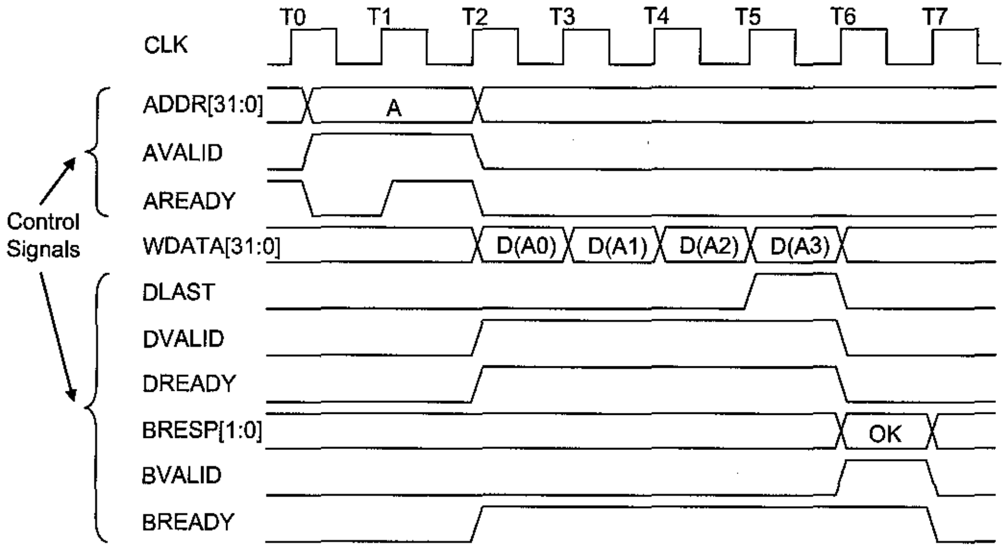


Figure 4.11: Transaction-based on-chip communication (Simplified AXI burst-write transaction).

4.4.3 IP Core Model for Æthereal NoC

The I/O ports of an IP core under test consist of primary inputs (PI), primary outputs (PO), scan chain inputs (SI) and scan chain outputs (SO). The PI's can be categorized into *primary data input* (PDI) and *primary control input* (PCI). Assuming that the CUT communicates with the NoC by means of the AXI protocol [91] described in Figure 4.11, the PDI consists of WDATA[31:0] signals, while PCI consists of ADDR[31:0], AVALID, DLAST, DVALID, and BREADY signals. The PO's can also be categorized into *primary data output* (PDO) and *primary control output* (PCO). PDO is made up of RDATA[31:0] signals (not included in the write transaction diagram in Figure 4.11), while PCO is made up of AREADY, DREADY, BRESP[1:0], and BVALID signals.

With the new classifications, core I/O's can be categorized as PDI, PDO, PCI, PCO, and other PI/PO's (PI'/PO'), which are not connected to the communication port of the NoC as shown in Figure 4.12. The PDI port is used to carry the test stimuli from the ATE to the CUT, and the PDO port is used to transport the test responses from the CUT to the ATE. The PCI and PCO control signals

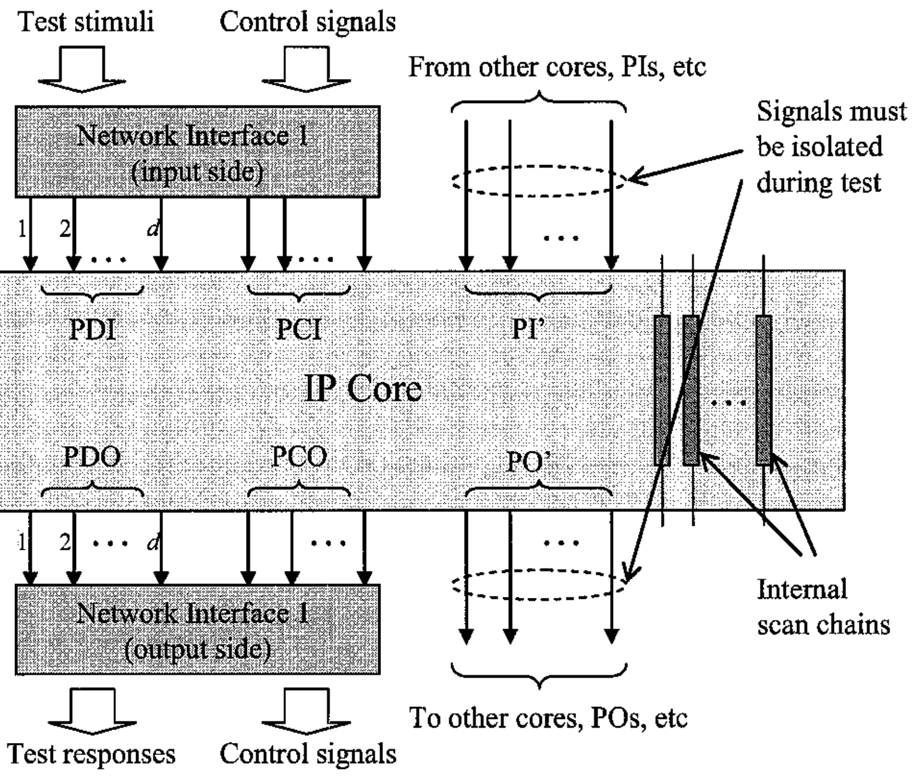


Figure 4.12: IP core model with an interface to the NoC's network interface and to other SoC cores and interconnects.

are needed to operate in the functional mode during the test application to ensure that the read/write transactions, by which the test data and responses are transmitted, execute properly.

During the test application, the NoC operates normally while the CUT is in test mode. Therefore, all the IP core inputs and outputs need to be isolated from external sources. This is one of the functions provided by the core wrapper. Since the CUT is not operating in the normal mode, the PCO signals must be generated by a wrapper controller and fed to the output side of the control signals. These special requirements during test are some of the issues that must be handled by the test wrapper. This is further discussed in Chapter 5, especially in Section 5.5.1.

Chapter 5

NoC-compatible Wrapper Design and Optimization

5.1 Introduction

As discussed in Chapter 1, the use of System-on-Chip (SoC) design methodology is prevalent in the electronics industry because of its many advantages. SoC enables previously-designed mega blocks to be reused in new designs with ease. With the aid of the efficient SoC design methodology, the time it takes to design all the way to ship a complex the product such as today's multifunction mobile devices can be as short as only six months. In such complex designs, most mega blocks such as processors, memories, memory controllers, I/O controllers, and typically reused from previous designs or licensed from external sources. The plug-and-play design style can even allow an individual chip designer to complete a relatively large design in a relatively short time.

While the progress in designs has been rather extraordinary, the same cannot be said for tests. The rapid increase in design complexity of SoC devices makes the manufacturing tests of such complex chips increasingly complex and expensive. A similar plug-and-play capable core-test methodology is needed in order for the VLSI tests cost to keep up with the design cost. The best-adopted core-based test technology for SoC's is based on the use of a Test Access Mechanism (TAM) [12, 21, 93] to connect all the embedded Cores-Under-Test (CUT) to the external Automatic Test Equipment (ATE).

Core-based tests require that the CUT's be isolated; this is typically achieved by wrapping the cores with IEEE 1500 [4] compatible wrappers. Various core test scheduling methodologies based on the dedicated TAM have been proposed [6, 12–28].

Chapters 2 and 3 explain some of the research related to this topic, including our works on test wrapper design and scheduling for SoC's that are based on a flat bus architecture and a hierarchical bus architecture. In this chapter, we look at the test challenges for the SoC which utilizes the Network-on-Chip interconnect in place of the bus-based interconnects. In [73], the authors discussed the test-related challenges that must be overcome when dealing with the NoC as an SoC interconnect.

This chapter focuses on the NoC-reuse-based wrapper architecture [94, 95]. We analyze two types of NoC-compatible wrappers, based on the guaranteed bandwidth and latency of the NoC. The first wrapper, Type 1, is based on the NoC-compatible wrapper proposed in [7, 96]. Depending on the number of wrapper scan chains, the test application time (TAT) of the Type 1 wrapper could be shorter or longer than that of the IEEE Std. 1500 wrapper; however, for most cases some NoC bandwidth is wasted. We then propose a second NoC-compatible wrapper, Type 2, that is 100% bandwidth efficient—i.e. no wasted bandwidth; Type 2's TAT is the same as that of the 1500 wrapper. For a given bandwidth or test application time constraint, the proposed wrapper optimization algorithm finds the best configuration using a fast binary search algorithm. Compared to [7, 96], our proposed test wrapper with the optimization scheme is more efficient in terms of both reducing the test application time and NoC bandwidth utilization; this is demonstrated by the experimental results reported in this chapter.

We begin with a review of some related work in Section 5.2. The NoC model and the IP core model are described in Section 5.3. Section 5.4 explains briefly the characteristics of the standard IEEE 1500 wrapper. Based on the 1500 wrapper limitations, Section 5.5 elaborates the proposed NoC-compatible wrapper architecture, which overcomes the weaknesses of the standard wrapper. Subsequently, the wrapper optimization methodology is explained in Section 5.6. Some experimental results on selected benchmark circuits are given in Section 5.7. Finally, concluding remarks are offered in Section 5.8.

5.2 Related Work

In this chapter, we will consider the Æthereal [79, 80] NoC, as an example, which provides abundant communication resources. Therefore, the use of a dedicated TAM for testing of an NoC-based chip is expensive. As a result, the reuse of functional on-chip resources for test purposes is becoming more practical and more economical. Several research groups have published work on NoC test scheduling [62, 63, 97] utilizing the NoC as the delivery path for the transport of the test data from external tester to the CUT's. Test scheduling for the NoC router [63, 98] and crosstalk test of the interconnects [99] have also been discussed. In these approaches, each CUT is wrapped by an IEEE 1500 compatible wrapper in order to provide isolation and access during the test application. With regard to the NoC's Design-for-Testability (DFT), the authors in [100] presented an architecture called ANoC-TEST, which targets the Asynchronous Networks-on-Chip (ANoC) [89].

IEEE 1500 standard wrapper relies on the use of a dedicated TAM, which merely provides an electrical connection between the wrapper and an external tester. When reusing the Networks-on-Chip (NoC) as TAM, the 1500 wrapper cannot be used as is because of three main reasons.

(i) The packet-based data transfer through the NoC cannot guarantee the precise timing required by the IEEE 1500 standard wrapper. At every scan clock, the 1500 wrapper requires that a new data is available at its scan input.

(ii) The wrapper does not provide the necessary control signals for both protocol inputs and outputs for successful packet transfer through the NoC. The 1500 wrapper was designed with the dedicated TAM in mind, while the control signals are provided through dedicated control lines parallel with the TAM.

(iii) The test data are transferred through a fixed-width data channel, which may result in wasted bandwidth and increased test application time. The 1500 wrapper assumes that the TAM width follows the number of wrapper scan chains.

In order to overcome these limitations, Amory *et al.* [7, 96] propose an NoC-compatible wrapper and controller that takes advantage of the guaranteed bandwidth and latency provided by the NoC to ensure test data integrity; this is achieved by using an input interface architecture that interfaces the NoC with the core. Their experimental results showed that in terms of core test time, the proposed NoC-compatible wrapper is comparable to the dedicated TAM-based IEEE 1500 wrapper, while having the advantage of being NoC-reuse [62, 63, 97–99] capable. However, due to the constraint of the parallel-serial conversion at the input port, the proposed wrapper requires much higher guaranteed bandwidth on the NoC than the actual rate of the test data loaded into the wrapper scan chains. This is further explained in Section 5.5.2.

5.3 NoC Reference Model

The proposed wrapper does not require a specific NoC architecture. Rather, the wrapper requires and utilizes the following characteristics of the NoC communication channels.

- The packet-switched communication channel can be divided into sub-channels, each with its own bandwidth capacity, which in a sense is equivalent to the TAM width. The wrapper uses a subset of the link bandwidth between the external I/O port and the embedded IP core by means of a bandwidth reservation scheme explained in Section 4.4.2.
- The wrapper uses the IP core's data ports for the transportation of the test stimuli and responses. Since the NoC operates in the functional mode during the test application, the wrapper needs to take into account the NoC capacity and functional control sequences.

The wrapper utilizes the functional communication channel between a tester and a CUT as a functional test access mechanism (TAM). The delivery channel can be a dedicated path or a transparent virtual channel. The quality-of-service guarantee through the bandwidth reservation scheme ensures that the test data are available at the CUT at the right time. To ease description, the Æthereal [79, 80] NoC described in Section 4.4.2 is used as a model to describe the characteristics and functionality of the proposed wrapper design and optimization.

The I/O ports of an IP core under test consist of primary inputs (PI), primary outputs (PO), scan chain inputs (SI) and scan chain outputs (SO). The PI's can be categorized into *primary data input* (PDI) and *primary control input* (PCI). With

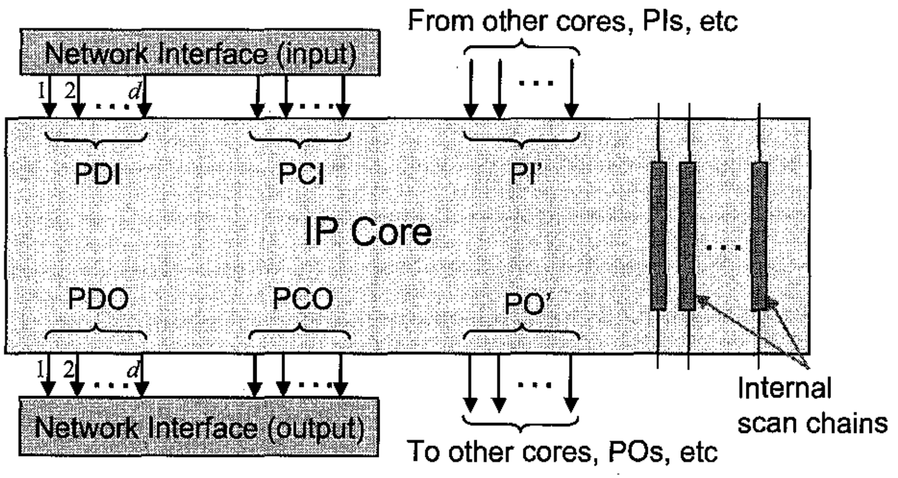


Figure 5.1: IP core model with an interface to the NoC's network interface and to other SoC cores and interconnects.

the new nomenclature described in Section 4.4.3, core I/Os can be categorized as PDI, PDO, PCI, PCO, and other PI/PO's (PI'/PO') which are not connected to the communication port of the NoC as shown. This is illustrated in Figure 5.1. The PDI's and PDO's are used to carry the test vectors from the ATE to the CUT, and the test responses from the CUT to the ATE, respectively. The PCI's and PCO's are needed to operate in the functional mode during the test application to ensure that the read/write transactions, by which the test data and responses are transmitted, execute properly.

Since the CUT is not operating in the normal mode, the PCO signals must be generated by a wrapper controller. Special wrapper cells proposed in [96] are used for PCO's to make the NoC operate in the normal mode to transfer the test responses. This is further discussed in Section 5.5.1. For all other PI/PO's, the standard IEEE 1500 wrapper boundary register cells are used.

5.4 IEEE Std. 1500 Wrapper Architecture

Core wrapper design for a dedicated TAM-based test architecture has been explained in [12, 21, 93]. For a CUT, given k internal scan chains (ISC) of length,

$l_1, l_2, ..., l_k$, i primary inputs, o primary outputs, b bidirectionals, and n_{sc} wrapper scan chains (WSC), the WSC's are formed while minimizing the maximum scan-in and scan-out depths. *Scan-in elements* consist of zero or more inputs, bidirectionals, and ISC's. *Scan-out elements* consist of zero or more outputs, bidirectionals, and ISC's.

Figure 5.2 shows $n_{sc} = 3$ for a CUT with $l_k \in [7, 5, 5, 3, 2]$ flip-flops, $i = 11$ ($n_{pdi} = 8$, $n_{pci} = 2$, $n_{pi'} = 1$), $o = 10$ ($n_{pdo} = 8$, $n_{pco} = 2$), and $b = 0$; in this chapter, n_x denotes the number of "x" elements. The scan elements are partitioned to form scan chains with *maximum scan-in depth*, $s_i = 11$, and *maximum scan-out depth*, $s_o = 11$, respectively. The wrapper scan chain formation treats the wrapper cells, regardless whether they are data or control I/Os, as identical. The scan chains are formed by cascading input cells, internal scan chains, and output cells together, in the specified order. As a result, the total test application time (TAT) can be calculated by Equation (5.1) [93], where n_v is the number of test vectors. For Figure 5.2, the TAT is, $T_{TAM} = 12n_v + 11$ clock cycles.

$$T_{TAM} = (max\{s_i, s_o\} + 1) \times n_v + min\{s_i, s_o\} \tag{5.1}$$

When using dedicated TAM's as the delivery channel, the wrapper scan chain inputs—either parallel inputs (WPI) and outputs (WPO), or serial inputs and outputs—are connected directly to the ATE input and output channels by means of the dedicated TAM wires. The functional I/O connections (dotted lines in Figure 5.2) are not used during testing. These functional I/O's are isolated from the CUT when the wrapper boundary register cells are configured in the test mode. The wrapper instruction register is used to enable test mode (solid lines) or the normal functional mode (dotted lines).

The current architecture of the standard IEEE 1500 wrapper has no capability to accept test stimuli from the functional input pins (i.e. pi'[0], pci[..], and pdi[..] pins). Neither does it have the ability to deliver test response data through the functional outputs (i.e. pco[..] and pdo[..] pins). Therefore, in this chapter, we explain how the propose wrappers address these issues when using the functional inputs and outputs during the core-based test application.

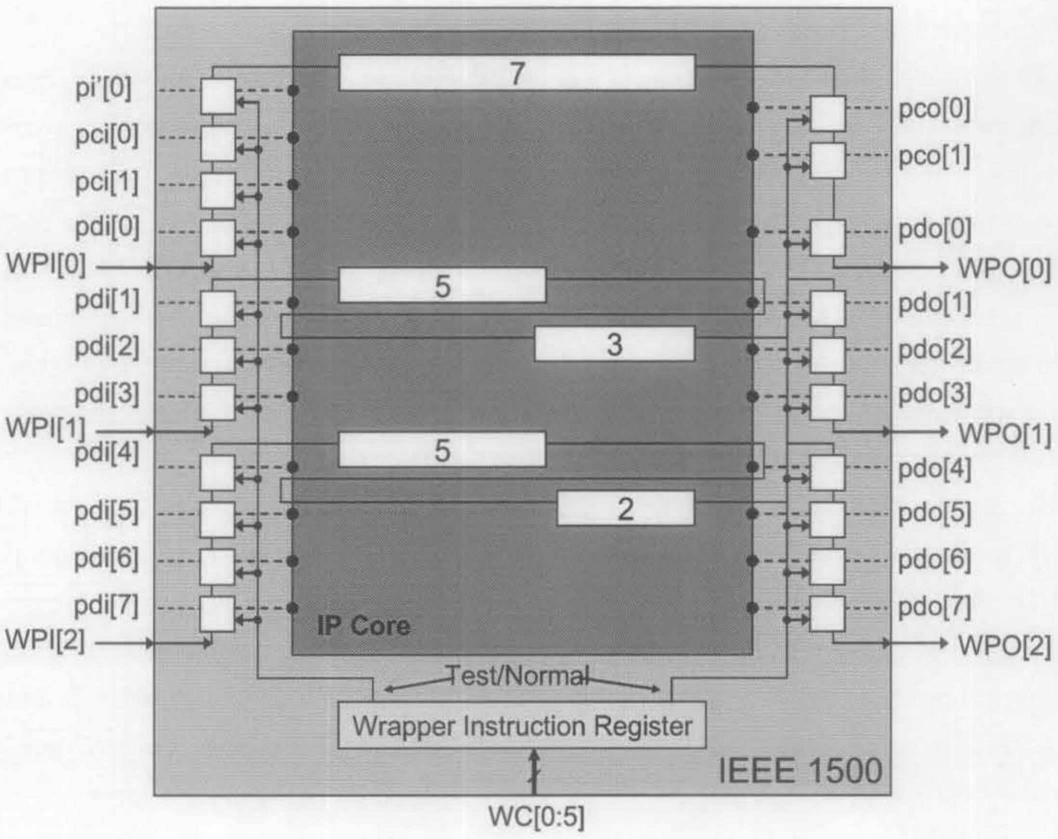


Figure 5.2: IEEE 1500 based wrapper with scan chains made up of PI/PO wrapper cells (squares) and internal scan chains (rectangles).

5.5 NoC-compatible Wrapper Architecture

In order to reuse the NoC as the delivery channel, the wrapper scan chains receive the test stimuli from the functional data inputs that are connected to the existing functional interconnects such as NoC interconnect. Without the dedicated TAM, the test data can no longer be transfered through the wrapper's test inputs. This requires the wrapper to capture the n-bit test data from the n-bit data port. Since the bit width of the wrapper scan chains, n_{sc}, are configured to optimize the test application time, n_{sc} does not always equal n. This bit width mismatch necessitates the use of a bandwidth matching mechanism at the core wrapper inputs as well as outputs. This mismatch is one of the major problems that the

NoC-compatible wrapper need to be able to accommodate.

Another problem arises because the mismatch in the modes of operations at different parts of the chip. The NoC interconnect, including the interfaces operates in the functional mode. It provides the functional data transfer services to the circuit under test (CUT), while the CUT's are tested—hence in the test mode. The handshake data transfer protocol such as OSI and OCP rely on the interactions between the master and the slave devices. The NoC requires some acknowledging signals from the CUT, which will not be able to provide the necessary signals in the test mode. Therefore, the NoC-compatible wrapper must be able to perform this task on behalf of the CUT during test.

Unlike the dedicated TAM wires, NoC's router-to-router interfaces operate based on the packet-switching scheme. There is no direct control from an external port to the CUT. Therefore, the test control and synchronization are no longer at the hand of the external automatic test equipment (ATE), rendering the IEEE 1500 wrapper inadequate. This is partly due to the inherent delay in packet-based data transfer used by the NoC. Sections 5.5.1–5.5.3 explain these problems and how they are addressed in the proposed NoC-compatible wrappers.

5.5.1 Type 1 NoC-compatible Wrapper: Interfacing the PDI/PDO Ports to the Scan Chains

The proposed Type 1 wrapper is the same as the wrapper in [96] in terms of wrapper boundary cells and scan chain structure. However, their operations are slightly different when loading the test stimuli into the wrapper scan chains. As a result the wrapper controllers are slightly different. We will discuss the effect of this characteristic on the test application time at the end of this section. The Type 1 wrapper uses the same approach as in [12, 21] when forming the wrapper scan chains which minimizes $max\{s_i, s_o\}$, except that most of the PDI and PDO cells are excluded from the scan chain formation.

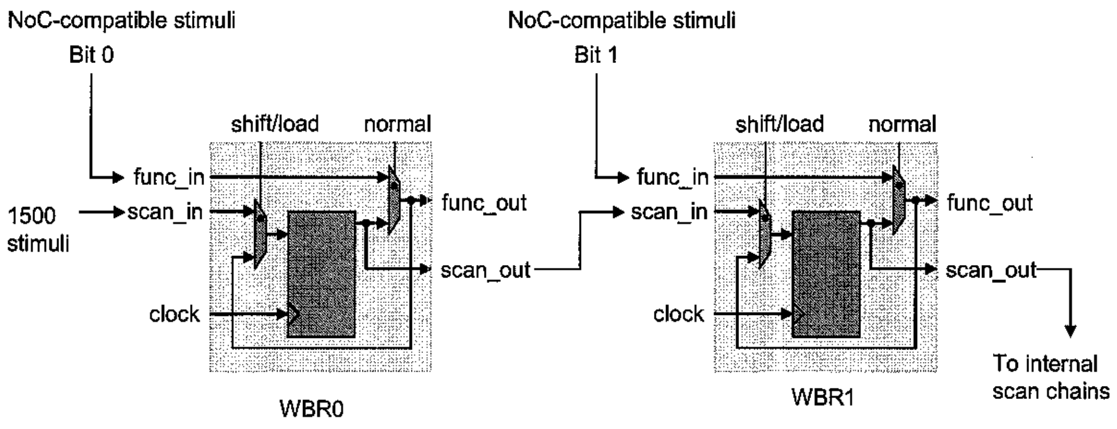
Figure 5.3 shows two variations of the wrapper scan chain architectures. The wrapper scan chains consist of two input and output wrapper cells each, and an 8-bit internal scan chain. In Figure 5.3(b) the test stimuli enters the scan chain through the shift input of the input wrapper cell (black squares). At every shift clock, one bit of test stimuli is shifted into the scan chain. As a result, it takes ten clock cycles to shift in the test stimuli and another ten cycles to shift out the test response.

Figure 5.3(a) illustrates how the 1500 stimuli is first shifted into the WBR0 in the first clock cycle. In the second shift clock, the first bit moves from WBR0 to WBR1 through the scan_out of WBR0 to scan_in of the WBR1. At the same time, WBR0 takes the second data bit. In the subsequent clock cycle, the first bit moves from WBR1 to the first bit of the internal scan chain. This scan-in process continues for ten clock cycles to fill in the test data into the WBR0, WBR1, and the internal scan chain.

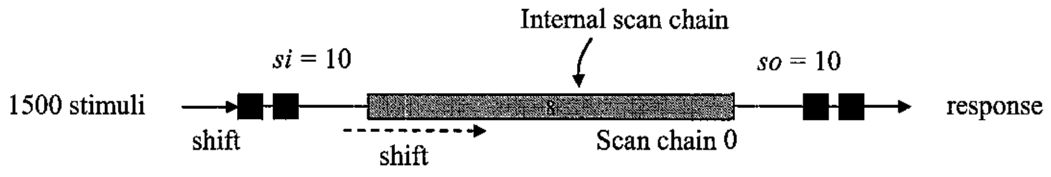
The NoC-compatible wrapper scan chain shown in Figure 5.3(c), on the other hand, takes the test stimuli from the functional data input (func_in) of the input wrapper boundary register cells instead of the scan input (scan_in). First the "shift/load" input toggles to load and "normal" input enabled. When clock is asserted, the 2-bit NoC-compatible stimuli (Bit 0 and Bit 1) are captured into WBR0 and WBR1, respectively. In the second stage, "shift/load" input toggles to shift mode. When clock is asserted, Bit 1 moves to the first bit of the internal scan chain and Bit 0 moves from WBR0 to WBR1. After two clock cycles, Bit 0 and Bit 1 are scanned into the internal scan chains. This process repeats four more times to shift the stimuli into the 8-bit internal scan chains.

During this process, the internal scan chains remain in the shift mode (i.e. test mode). Only the WBR cells are required to switch between shift and load modes to enable the bandwidth matching process between the 2-bit data input and 1-bit wide wrapper scan chain. Based on this scan-in procedure, the effective scan-in depth of the NoC-compatible wrapper scan chain is eight, which excludes the wrapper boundary register cells. Next we formulate the general conditions for an arbitrary bit widths between the NoC data port and the wrapper scan chains.

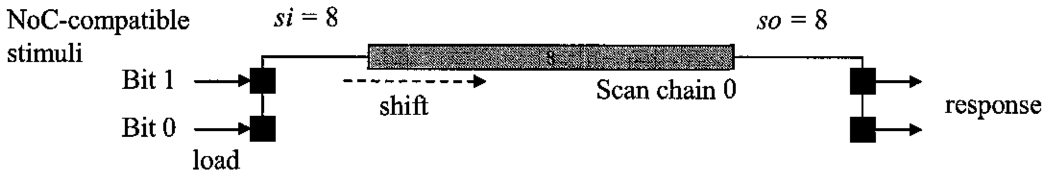
For a given number of wrapper scan chains, n_{sc}, and the PDI bit-width, n_{pdi}, the number of PDI bits that can be used to carry the test data for each wrapper scan chain, n_{idwc}, is given by Equation (5.2), assuming that $n_{pdi} \geq n_{sc}$. To differentiate these PDI bits, those that can carry the test data are called *input data wrapper cells*, IDWC (shaded black in Figure 5.4). If $\hat{n}_{idwc} \neq 0$ (Equation (5.3)), some PDI bits cannot be used to carry the test data; these will become



(a) Wrapper boundary register architecture.



(b) IEEE 1500 scan chain architecture.



(c) NoC-compatible scan chain architecture.

Figure 5.3: Components of a wrapper scan chain which takes test stimuli from the functional data port. One 8-bit internal scan chains, 2-bit data input, and 2-bit data output ports.

part of the wrapper scan chains, and not the IDWC. A similar analysis can be done for the *output data wrapper cells* (ODWC), resulting in equations (5.4) and (5.5).

$$n_{idwc} = \lfloor n_{pdi}/n_{sc} \rfloor \tag{5.2}$$

$$\hat{n}_{idwc} = n_{pdi} \; mod \; n_{sc} \tag{5.3}$$

$$n_{odwc} = \lfloor n_{pdo}/n_{sc} \rfloor \tag{5.4}$$

$$\hat{n}_{odwc} = n_{pdo} \; mod \; n_{sc} \tag{5.5}$$

The Type 1 NoC-compatible wrapper is illustrated in Figure 5.4 for the CUT with 8-bit PDI/PDO's, 2-bit PCI/PCO's, 1-bit PI', and three wrapper scan chains (refer to the notation in Figure 5.1). From Equation (5.2), $n_{idwc} = n_{odwc} = \lfloor 8/3 \rfloor = 2$ means that each wrapper scan chain is interfaced to two IDWC/ODWC cells. In addition, $\hat{n}_{idwc} = \hat{n}_{odwc} = (8 \; mod \; 3) = 2$ means that the remaining two PDI/PDO bits cannot be used to carry the test data (illustrated by the dotted lines for pdi[0]* and pdi[5]*); these unused PDI/PDO bits become part of the wrapper scan chain, with no extra functionality. In the figure, dotted lines represent the functional paths which are not used during the scan operation. Solid lines represent the test data (stimuli and responses) transportation paths during the scan-in/out operations.

The 2-bit input control signals (PCI) coming from the NoC are used by the controller to synchronize the load and shift control signals required in order to capture the test data from the PDI inputs into the corresponding IDWC cells and scan chain elements. Since the wrapper cells for PCI inputs are always in scan mode during the test application, the incoming signals are ignored by the CUT. Similarly, the control signals coming from the CUT (PCO) are ignored by the wrapper cells because the generated signals are invalid during the test operation. Instead, similar to the scheme proposed in [96], the controller must generate the necessary control signals (Figure 4.11) and feed them to the NoC through the special wrapper cell at each PCO output, which is illustrated in Figure 5.5. These functional control signals are necessary to ensure successful data transfer through the NoC in the functional mode.

Since n_{pdi} equals n_{pdo} for a typical NoC core, the following discussion on the PDI on the input port also applies to the PDO on the output port. During the

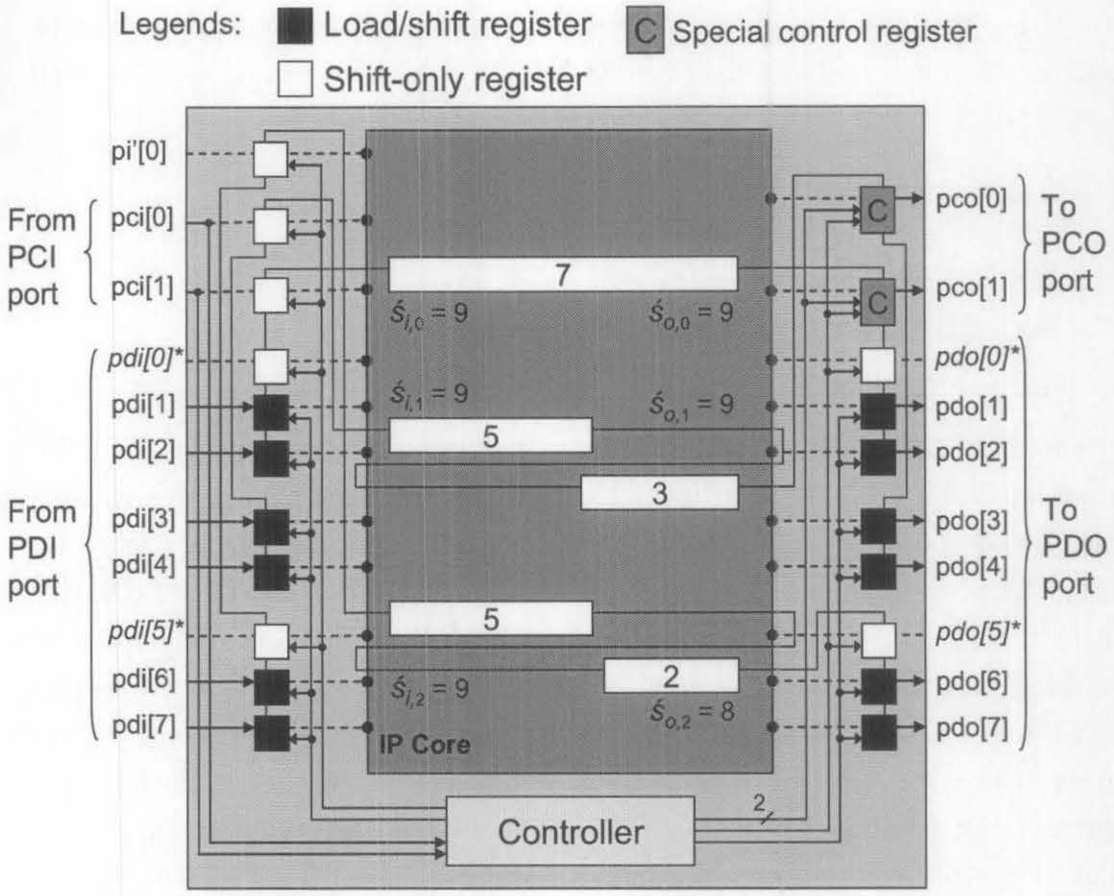


Figure 5.4: Type 1 NoC-compatible wrapper architecture with scan chains made up of internal scan chains and normal (shift-only) wrapper cells in Figure 5.5.

test application, IDWC cells are loaded with the test data in one clock cycle, in the normal operation mode (refer to Figure 5.4). The IDWC cells change into the test mode, during which the test data are serially shifted for two clock cycles to empty the contents into the scan chains. After completion, the IDWC cells change again into the normal mode to capture the next incoming data from the PDI port. This operation is controlled by a test controller which keeps track of the number of loads and shifts using counters (Figure 2.8(b)).

For the NoC-compatible wrapper with a scan-in depth of nine (Figure 5.4), after four repetitions of loads and shifts, the first eight bits of each scan chains

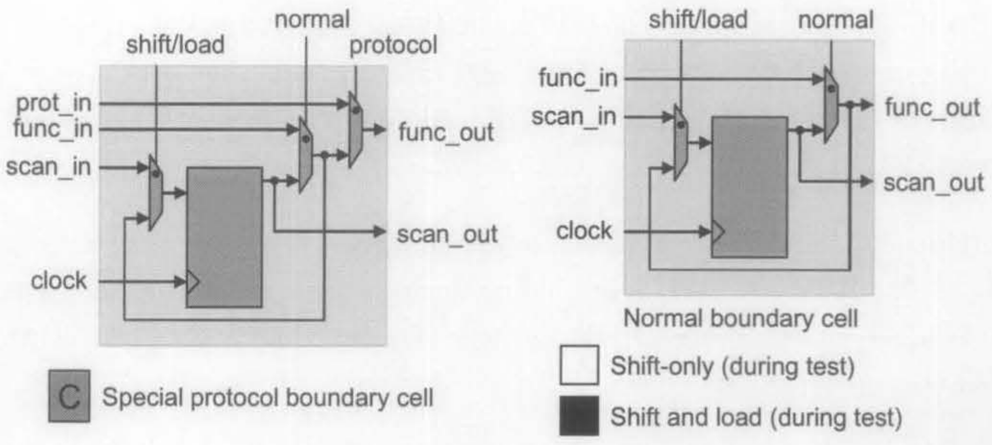


Figure 5.5: Two types of wrapper cells used in [96]. The same normal wrapper cell is used under different control sequences for data-capturing from the NoC (black), and as part of the wrapper scan chains (white). The special wrapper cell has an extra *prot_in* input signal that bypasses the memory cell to supply the functional (i.e. bus protocol) control signals to return the test response data through the NoC.

are loaded with the test data. To load the last bit, the IDWC cells are loaded with new test data and a single shift clock is applied. However, before applying the capture cycle, the IDWC must also be loaded with valid test data. After the last single shift, only part of the IDWC cells contains valid test data. Reloading the IDWC data from the PDI port can corrupt the valid data currently in the IDWC cells. The wrapper control scheme in [7, 96] does not take into account this possible data corruption during the scan operation.

To overcome this problem, the first (s_i mod n_{idwc}) shift cycles of every test pattern must shift in dummy bits into the scan chains followed by the load-shift cycles until all scan chain elements are filled with test vector data. After the scan chains are completely loaded, another clock cycle is required to load the IDWC cells with valid test data before applying the capture cycle. Since the IDWC and ODWC wrapper cells are not considered part of the wrapper scan chains, the effective scan-in elements for the proposed wrapper scan chain design can be formally defined as follows.

Definition 5.1 The *scan-in elements of the Type 1 NoC-compatible wrapper* consist of the unused IDWC cells, bidirectional cells, and internal scan chains (i.e. excluding all the other IDWC cells). The maximum scan-in depth is denoted by $\acute{s}_i$ (Figure 5.4). ∎

Definition 5.2 The *scan-out elements of the Type 1 NoC-compatible wrapper* consist of the unused ODWC cells, bidirectional cells, and internal scan chains (i.e. excluding all the other ODWC cells). The maximum scan-out depth is denoted by $\acute{s}_o$ (Figure 5.4). ∎

As a result of the new test scheme, the number of shift-in and shift-out cycles required for the Type 1 NoC-compatible wrapper is summarized by equations (5.6) and (5.7), respectively. Equation (5.8) gives the total TAT, where the additional "+1" represents the final load of the IDWC data prior to the capture cycle. For the NoC-compatible wrapper in Figure 5.4, $T_{Type1} = 11n_v + 9$ clock cycles, which is smaller than T_{TAM} based on Equation (5.1). The reduction in TAT is due to the IDWC and ODWC cells that are not part of the wrapper scan chains. The IDWC cells are loaded in parallel instead of through serial shifting.

$$\widehat{s_i} = \acute{s}_i + (\acute{s}_i \ mod \ n_{idwc}) \tag{5.6}$$

$$\widehat{s_o} = \acute{s}_o + (\acute{s}_o \ mod \ n_{odwc}) \tag{5.7}$$

$$T_{Type1} = (max\{\widehat{s_i}, \widehat{s_o}\} + 1 + 1) \times n_v + min\{\widehat{s_i}, \widehat{s_o}\} \tag{5.8}$$

5.5.2 Type 1 NoC-compatible Wrapper: Inefficient NoC Bandwidth Utilization

For a CUT with n_{sc} wrapper scan chains and f_m scan frequency, its scan rate (or scan bandwidth) is given by $B^{scan}_{Type1} = n_{sc} \times f_m$. As shown in the previous example (Figure 5.4), some PDI bits cannot be used to carry the test data due to the Type 1 wrapper's input architecture constraint. In order to supply the test data to the CUT at B^{scan}_{Type1} rate, the required channel bandwidth on the NoC is given in Equation (5.9). For the NoC-compatible wrapper in Figure 5.4, the scan and required bandwidths are $3f_m$ bits-per-second (*bps*) and $4f_m$ *bps*, respectively.

$$B^{req}_{Type1} = B^{scan}_{Type1} \times \frac{n_{pdi}}{n_{pdi} - \hat{n}_{idwc}} \tag{5.9}$$

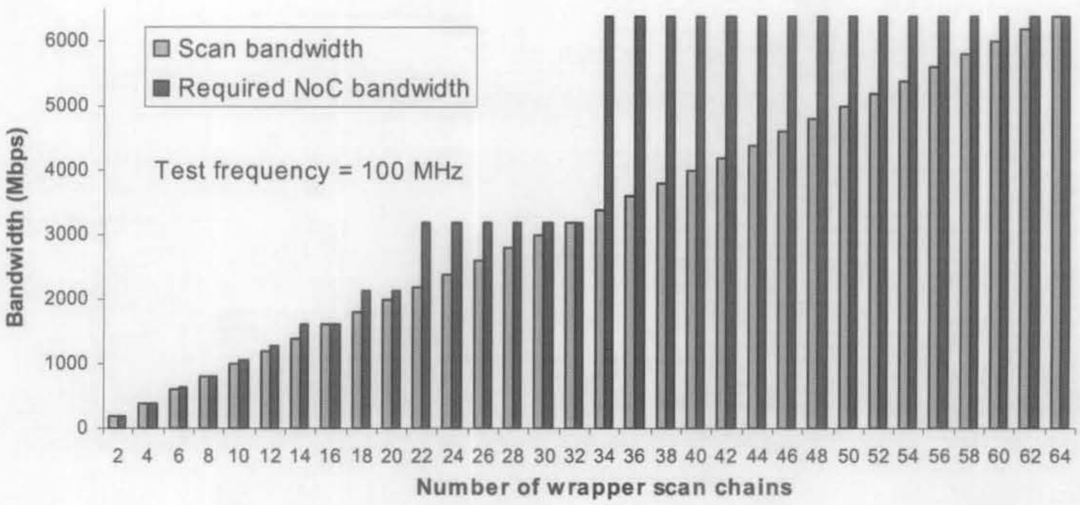


Figure 5.6: Scan rate and required bandwidth of a Type 1 NoC-compatible wrapper for p93791's Core 6 [1] with $n_{pdi} = 64$.

Figure 5.6 shows the required bandwidth of the proposed Type 1 NoC-compatible wrapper (Figure 5.4) compared to the actual scan bandwidth for an ITC'02 benchmark circuit for $n_{sc} = 2$ to 64 and $n_{pdi} = 64$. For some number of wrapper scan chains, the required bandwidth is almost twice that of the scan bandwidth. For these cases (i.e. $\hat{n}_{idwc} \neq 0$), the Type 1 NoC-compatible wrapper is inefficient in terms of NoC bandwidth utilization, similar to the NoC-compatible wrapper in [96]. For other cases, it is as efficient as the dedicated TAM-based wrapper while having the advantage of NoC reuse support capability with minimal area overhead. In the next section, an alternate wrapper architecture is proposed to overcome this limitation.

5.5.3 Type 2 NoC-compatible Wrapper: Optimizing the NoC Bandwidth Utilization

Section 5.5.2 has shown that the Type 1 wrapper is inefficient in terms of bandwidth utilization because of the restricted input/output wrapper cells architecture. The Type 2 NoC-compatible wrapper in Figure 5.7 is designed to comple-

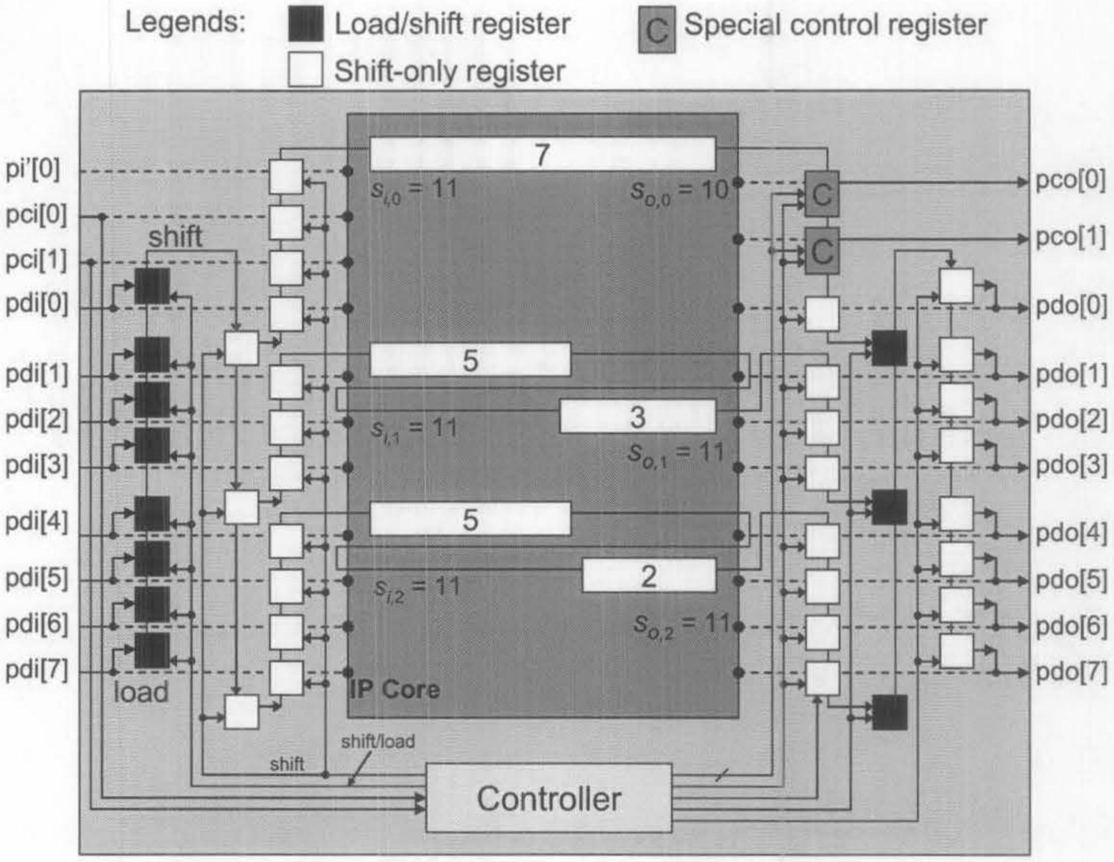


Figure 5.7: Type 2 NoC-compatible wrapper with an I/O interface which performs parallel-serial shifting to match the NI bit width with the number of wrapper scan chains. The same wrapper cells in Figure 5.5 are used.

ment the Type 1 wrapper in this aspect. Extra load/shift registers and shift-only registers are added to the PDI/PDO ports, similar to the bandwidth matching output register architecture in Figure 2.8(a) for the reuse of the SoC's functional bus; the concept of bandwidth matching architecture is first proposed in [101]. On the input side, the load/shift registers translate the PDI bit-width into the number of wrapper scan chains using parallel-serial shift registers.

In this chapter, we distinguish the terms *load, shift* and *scan* as follows.

Definition 5.3 *Load* operation captures data into the wrapper boundary register

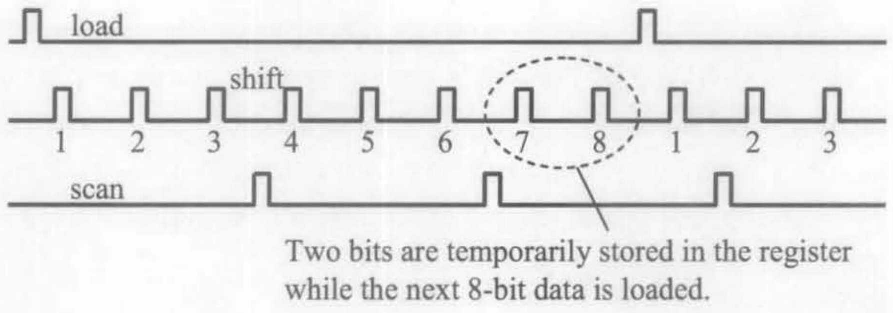

load
shift
1
2
3
4
5
6
7
8
1
2
3
scan
Two bits are temporarily stored in the register while the next 8-bit data is loaded.

Figure 5.8: Control signals sequence of the Type 2 wrapper to perform the bit width translation. Bits 7 and 8 from the first load are temporarily stored in the shift-only buffer while waiting for the first bit of the next load.

(WBR) from its data input while shift and scan operation takes data from the shift input. Furthermore, *shift* operation takes place along the bandwidth-matching WBR chain consisting of the load/shift registers and the additional shift-only registers that are not part of the wrapper scan chains. *Scan* operation takes place along the wrapper scan chains as in Figure 5.2. ∎

Control signals on pci[0:1] indicate new data availability at the pdi[0:7] port, which triggers the Controller to assert a load signal to capture the data into the load/shift registers. Subsequently, the Controller asserts a shift signal on the load/shift registers and the 3-bit shift-only registers for $n_{sc} = 3$ cycles. This is followed by a scan signal on the wrapper scan chain (Figure 5.8). This process is repeated until all the data in the pdi[0:7] register is shifted out. As explained in Figure 5.8, bits 7 and 8 need to be scanned together with bit 1 of the next load cycle. The 3-bit shift-only register is necessary to store bits 7 and 8 while new data is loaded. Therefore, no NoC bandwidth is wasted. When the capture clock is asserted, the 3-bit shift-only registers contain the data for the first scan cycle of the next test pattern. Therefore, they are not considered part of the wrapper scan chains.

As a result, all the PDI wires can be used to carry the test data; therefore, the required NoC bandwidth matches the scan bandwidth for any wrapper configuration. The TAT for the Type 2 NoC-compatible wrapper is also the same as the dedicated TAM-based wrappers, given in Equation (5.10). This is achieved at the cost of area overhead of load/shift registers and a more complex control scheme to realize the bit-width conversion. Therefore, it is important that the Type 2 wrapper is used only when necessary. Section 5.6 looks at two proposed optimization schemes for both of these NoC-compatible wrappers.

$$T_{Type2} = (max\{s_i, s_o\} + 1) \times n + min\{s_i, s_o\} \tag{5.10}$$

5.6 Optimization of the NoC-compatible Wrappers

Parallel core tests are performed according to a test schedule under given constraints. Figure 5.9 shows an example test scheduling scheme based on the bin-packing optimization [12, 21], where a rectangle represents the required NoC bandwidth (vertical axis) and the TAT (horizontal axis) of a CUT under a specific wrapper configuration. The figure illustrates the state of the test schedule after four cores are scheduled (i.e. the starting test times and the amount of allocated bandwidths are assigned). When scheduling the subsequent core, there are several possible starting times and amount of bandwidths that can be assigned to the core. B_1 and B_2 are the maximum amount of bandwidths that can be allocated if the test were to begin after the test of Core 2 and Core 3, respectively, complete. Using B_1 and B_2 as inputs to the wrapper optimization algorithm Ψ_B (Problem 5.1 defined in the next paragraph), we can determine the length of the test application by maximizing the bandwidth utilization. Based on these information, we can decide how to schedule the subsequent core test. Similarly, we could also consider the available test time instead of bandwidth during the test scheduling. Nonetheless, we leave the problem of test scheduling based on the bandwidth allocation scheme as a subject of the next chapter.

Based on the above scheduling objectives, the problems of optimizing the number of wrapper scan chains (n_{sc}) for a core, under a given bandwidth (Ψ_B) or a given test application time (Ψ_T), respectively, can be formally defined as follows.

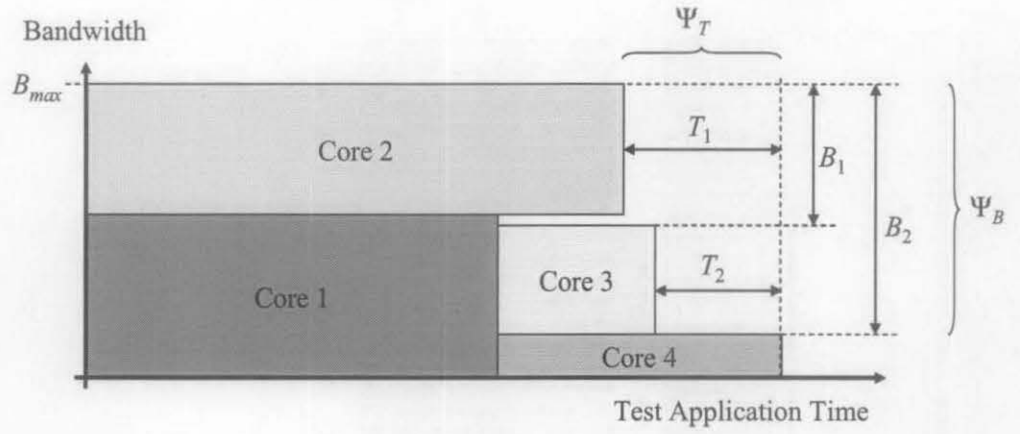


Figure 5.9: A typical test schedule optimization scheme based on 2D-bin packing algorithm.

Problem 5.1 [Ψ_B] Given a core with i functional inputs, o functional outputs, b bidirectionals, k internal scan chains of length $l_1, l_2, ..., l_k$, scan frequency, f_m, and a maximum bandwidth for the virtual channel between the core and the ATE, B_{max}, find the number of wrapper scan chains, n_{sc}, such that

(i) the TAT is minimized,
(ii) the required bandwidth, $B_{req} \leq B_{max}$, and
(iii) n_{sc} is minimum subject to priority (i). ■

Problem 5.2 [Ψ_T] Given a core as in Ψ_B, and a maximum TAT, T_{max}, find the number of wrapper scan chains, n_{sc}, such that

(i) the required bandwidth, B_{req}, is minimized,
(ii) TAT $\leq T_{max}$, and
(iii) n_{sc} is minimum subject to priority (i). ■

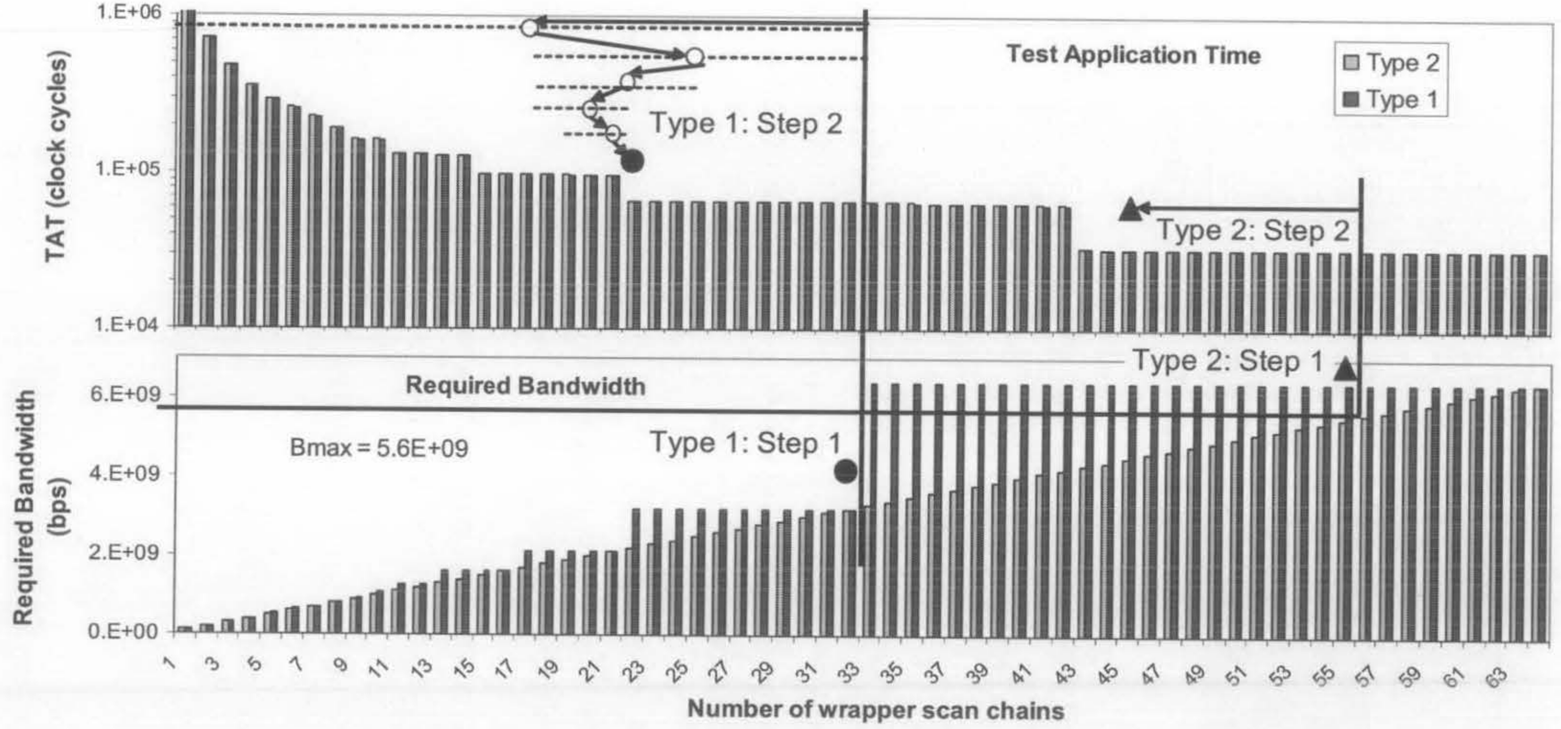


Figure 5.10: Optimization of NoC-compatible wrapper design for a given B_{max}. In Step 2 (Type 1), the dotted lines represent the search space which halves in every progression of the binary search.

A similar problem for a dedicated TAM-based wrapper design has been proved NP-hard in [12]. Therefore, heuristic algorithms are proposed to solve both Ψ_B and Ψ_T. Figure 5.10 illustrates graphically the search steps for Ψ_B (when $B_{max} = 5600$ Mbps) for Core 17 of the p93791 [1] benchmark circuit. Since the TAT and the required bandwidth are monotonic decreasing and increasing with respect to n_{sc}, respectively, binary search algorithms can be used to find the solution for n_{sc}. At each search step, wrapper scan chains which minimize $max\{s_i, s_o\}$ are formed using the algorithm proposed in [12], described in Section 5.5. For the Type 1 wrapper, binary search takes place in steps 1 and 2 (refer to Figure 5.10). In Step 1, the maximum number of scan chains, n_{sc}^{max}, such that $B_{Type1}^{req} \leq B_{max}$ is located (objective *(ii)* of Ψ_B). In Step 2, the search is restricted to $n_{sc} = [1, n_{sc}^{max}]$ to find the solution(s) for n_{sc} that minimizes the TAT (objective *(i)* of Ψ_B). Because of the staircase decreasing TAT vs. n_{sc} (top half of Figure 5.10), multiple solutions to n_{sc} may exist. The smallest value is chosen as the solution (objective *(iii)* of Ψ_B) without affecting objective *(i)*. Progression of the binary search is graphically illustrated in Figure 5.10. As a result, $n_{sc} = 22$ (Type 1) with a TAT of 65,098 clock cycles.

For the Type 2 wrapper, n_{sc}^{max} is directly calculated since B_{Type2}^{req} is a linear function of n_{sc}. Binary search in Step 2 (similar to the Type 1 wrapper) results in $n_{sc} = 45$ with a TAT of 32,766 clock cycles. Clearly a better result for the Type 2 wrapper when $B_{max} = 5600$ Mbps. In this case, the Type 1 wrapper is unable to utilize efficiently the allocated bandwidth because of the constraint in its I/O architecture. A similar heuristic is implemented for Ψ_T and some selected cases for both algorithms are presented in Section 5.7.

5.7 Experimental Results

In order to evaluate the effectiveness of the proposed methodology, we have conducted experiments on several benchmark IP cores. Core 17 and Core 6 (the largest of p93791 circuit) from the ITC'02 benchmark [1] are selected in order to offer comparisons with the IEEE 1500-based approaches reported in [12, 93]. Another IP core—an example core from [7]—allows some comparison with an NoC-compatible wrapper to be offered. Finally, we offer an extensive comparison with [96] using 42 different cores from ITC'02 circuits. The scan frequency is fixed to $f_m = 100$ MHz; the TAT reported in this chapter is in number of scan clock cycles, where each cycle is equivalent to $1/f_m$ or $0.01\mu s$.

A test application time (TAT) comparisons between the proposed Type 1 NoC-compatible wrapper (third column) and dedicated TAM-based IEEE 1500 wrapper (second column) are given in Table 5.1, for Core 6 with $n_{pdi} = 64$ bits. The percent (%) increase shown in the fourth column is calculated as ((column 3) - (column 2)) / (column 2) $\times$ 100%. In all cases, the differences are always less than 0.2%; the proposed Type 1 NoC-compatible wrapper does not incur noticeable penalty on the TAT. In fact, some reductions are achieved for $n_{sc} = 1$ and 2 scan chains.

For the Type 2 NoC-compatible wrapper, the TAT is the same as the dedicated TAM-based approach because the added interface between the CUT and the NoC port does not constrain the scan chain design. The Type 2 wrapper's required bandwidth matches the scan bandwidth—an improvement due to the extra load/shift registers. Table 5.2 reports similar experimental results for Core 17 of the same benchmark circuit. The % increase is calculated in the same way as that of Table 5.1. The TAT of the proposed NoC reuse wrapper is at most 0.66% larger than the standard wrapper.

For the circuit from [7], the TAT is given in Table 5.3. Compared to the dedicated TAM-based wrapper, the proposed Type 1 NoC-compatible wrapper is better for smaller number of wrapper scan chains. For wider scan chains, the TAT's are about 3% longer. However, compared to the NoC-compatible wrapper design in [7][1], the Type 1 wrapper is always superior, with about 4% shorter test application time. The % increase in columns labeled [Type 1 / 1500] and [Type 1 / Amory] are calculated as ([Type 1] - [1500]) / [1500] $\times$ 100% and ([Type 1] - [Amory]) / [Amory] $\times$ 100%, respectively.

Table 5.4 and Table 5.5 gives further comparison for Core 6 and Core 17, respectively, of the p93791 benchmark circuit. The TAT (column 2) and the required bandwidth, B^{req}_{Amory}, (column 3) are obtained for selected n_{sc} (column 1). Using $B_{max} = B^{req}_{Amory}$ (column 4) as input to Ψ_B, the corresponding n_{sc},

[1]Based on the corrected results obtained from the paper author because of reporting error in the original published literature.

Table 5.1: Test time comparison between standard 1500 wrapper and the proposed Type 1 wrapper for Core 6 of p93791 [1] with 64-bit PDI/PDO's.

	TAT (clock cycles)		
n_{sc}	[12, 93]	Type 1 NoC	% increase
1	5,317,007	5,312,372	-0.09
2	2,658,613	2,656,404	-0.08
3	1,809,815	1,812,442	0.15
4	1,358,456	1,359,988	0.11
5	1,126,316	1,127,848	0.14
6	907,097	909,286	0.24
7	793,217	794,749	0.19
8	679,337	680,212	0.13
9	674,957	676,489	0.23
10	565,457	566,770	0.23
11	561,077	562,171	0.19
12	455,738	455,956	0.05
13	451,577	452,452	0.19
14	451,358	451,576	0.05
15	447,197	448,070	0.20
16-19	341,858	342,076	0.06
20-21	337,478	338,134	0.19
22	333,317	333,754	0.13
23	231,478	231,258	-0.10
24-38	227,978	228,196	0.10
39-42	223,598	223,816	0.10
43-45	219,218	219,436	0.10
46	115,848	115,847	0.00
47-64	114,317	114,535	0.19

B^{req}_{Type2}, and TAT for the proposed Type 2 wrapper are obtained. Using at most the bandwidth required by [7], the proposed wrapper gives shorter TAT's. %

Table 5.2: Test time comparison between standard 1500 wrapper and the proposed Type 1 wrapper for Core 17 of p93791 [1] with 64-bit PDI/PDO's.

	TAT (clock cycles)		
n_{sc}	[12, 93]	Type 1 NoC	% increase
1	1,433,858	1,430,832	-0.21
2	717,148	715,648	-0.21
3	483,258	484,342	0.22
4	358,682	358,040	-0.18
5	290,128	290,778	0.22
6	257,361	257,577	0.08
7	225,028	225,244	0.10
8	192,912	193,128	0.11
9	161,664	161,880	0.13
10	160,579	161,229	0.40
11	130,628	130,196	-0.33
12	129,331	129,764	0.33
13	128,680	128,896	0.17
14	128,029	128,462	0.34
15-16	97,215	97,648	0.45
17	97,215	97,431	0.22
18	96,998	97,214	0.22
19	96,347	96,563	0.22
20	95,913	96,129	0.23
21	95,696	95,912	0.23
22	65,530	65,314	-0.33
23-34	64,882	65,098	0.33
35	64,665	64,881	0.33
36	64,448	64,664	0.34
37-38	64,014	64,230	0.34
39	63,797	64,013	0.34
40	63,363	63,579	0.34
41-42	63,146	63,362	0.34
43	33,632	33,632	0.00
44	32,982	32,982	0.00
45-64	32,766	32,982	0.66

Table 5.3: TAT comparison for the circuit defined in [7].

	1500	Amory	Proposed		% increase	
n_{sc}	[12, 93]	[7]	Type 1	Type 2	Type 1 / 1500	Type 1 / Amory
1	5,532	5,532	5,300	5,532	-4.19	-4.19
2	2,771	2,771	2,660	2,771	-4.01	-4.01
3	1,858	1,858	1,780	1,858	-4.20	-4.20
4	1,396	1,451	1,428	1,396	2.29	-1.59
5	1,363	1,429	1,406	1,363	3.15	-1.61
6	1,363	1,418	1,395	1,363	2.35	-1.62

Table 5.4: TAT comparison with [7] for Core 6 of p93791, with 64-bit PDI/PDO port.

[7]			Proposed (Type 2)					
							% increase	
n_{sc}	TAT	B_{req}	B_{max}	n_{sc}	B_{req}	TAT	B_{req}	TAT
11	562,172	1,280	1,280	12	1,200	455,738	-6.3	-18.9
15	448,073	1,600	1,600	16	1,600	341,858	0.0	-23.7
22	333,755	3,200	3,200	24	2,400	227,978	-25.0	-31.7
24	228,416	3,200	3,200	24	2,400	227,978	-25.0	-0.2

increase B_{req} in column 8 is calculated as ((column 6) - (column 3)) / (column 3) $\times$ 100%. % increase TAT in column 9 is calculated as ((column 7) - (column 2)) / (column 2) $\times$ 100%.

In Table 5.4, for n_{sc} = 11 scan chains (first row), the proposed wrapper requires 6.3% less bandwidth to obtain 18.9% smaller TAT, than the given wrapper configuration by the method in [7]. For the selected cases in the tables, the proposed approach either requires less bandwidth to achieve comparable TAT, or achieves smaller TAT while requiring similar amount of bandwidth.

Table 5.6 compares the Type 1 and Type 2 wrappers when Ψ_B and Ψ_T are

Table 5.5: TAT comparison with [7] for Core 17 of p93791, with 64-bit PDI/PDO port.

[7]			Proposed (Type 2)					
							% increase	
n_{sc}	TAT	B_{req}	B_{max}	n_{sc}	B_{req}	TAT	B_{req}	TAT
6	259,531	640	640	6	600	257,361	-6.3	-0.8
13	129,548	1,600	1,600	15	1,500	97,215	-6.3	-25.0
23	65,316	3,200	3,200	23	2,300	64,882	-28.1	-0.7
33	65,099	6,400	6,400	45	4,500	32,766	-29.7	-49.7

Table 5.6: Optimization results for selected values of B_{max} and T_{max} (Core 17 of p93791).

	Given:	Type 1			Type 2		
	B_{max}	n_{sc}	B_{req}	TAT	n_{sc}	B_{req}	TAT
Ψ_B	1,700	15	1,600	97,648	15	1,500	97,215
	3,000	21	2,133	96,129	23	2,300	64,882

	T_{max}	n_{sc}	B_{req} (Mbps)	TAT	n_{sc}	B_{req} (Mbps)	TAT
Ψ_T	70,000	22	3,200	65,098	22	2,200	65,530
	200,000	8	800	193,128	8	800	192,912

applied. For $B_{max} = 1700$ Mbps, both wrappers result in similar performance—a slight advantage for Type 1 in terms of area overhead. At $B_{max} = 3000$ Mbps, Type 2 is clearly the winner, with only 0.8% bandwidth overhead to achieve 32.5% TAT reduction. For $T_{max} = 70000$, Type 2 requires 31% smaller bandwidth with less than 0.7% TAT overhead. On the other hand, at $T_{max} = 200000$, Type 1 wrapper is superior due to its minimal wrapper hardware overhead. The results illustrate the tradeoffs between the two types of NoC-compatible wrappers for a given constraint, which can be explored during the test schedule optimization.

Table 5.7: List of ITC'02 benchmark cores included in the experiments reported in Figures 5.11 and 5.12.

SoC	Cores considered
a586710	2, 3, 4, 7
d281	7
d695	2
f2126	1, 2
g1023	1, 2, 3, 4, 10, 11, 12, 14
h953	1
p22810	27
p34392	2, 10, 18
p93791	10, 32
q12710	1, 2, 3, 4
t512505	1, 2, 4, 8, 9, 14, 15, 16, 17, 23, 24, 25, 26, 29, 31

An extensive comparison on area overhead and test application time between the proposed Type 2 wrapper and the wrapper in [96] is offered using 42 selected cores from the ITC'02 benchmark circuits shown in Table 5.7. Sixteen unique channel bandwidth values, $B_{max} \in [1 \times f_m, 2 \times f_m, ..., 16 \times f_m]$, in bits-per-second are considered. For every B_{max}, the wrapper configurations for both the Type 2 wrapper and the wrapper proposed in [96] are determined. The corresponding TAT and area costs in terms of wrapper boundary cells are given in Figs. 5.11 and 5.12. Figure 5.11 shows the area increase of the Type 2 wrapper, relative to that of the wrapper in [96]. The horizontal axis is the core ID number in the order listed in Table 5.7. The average relative area increase for the 672 wrapper configurations is 19.3%. Figure 5.12 gives the corresponding test time comparison, for the 16 different B_{max} values for each core. The proposed wrapper achieved up to 48.6% reduction and an average of 7.8% shorter test application time. For any given maximum channel bandwidth, B_{max}, Type 2's TAT is always shorter than that of [96].

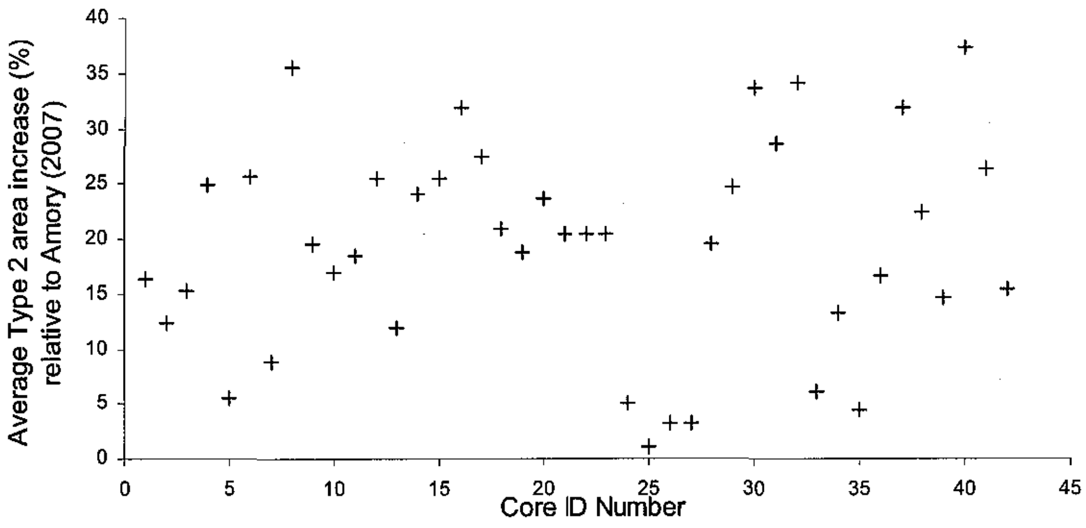


Figure 5.11: Area comparison between the Type 2 wrapper and the wrapper in [96]. The wrapper configurations are determined for 16 unique values of maximum bandwidth, $B_{max} \in [1 \times f_m, 2 \times f_m, ..., 16 \times f_m]$ *bps* and $n_{pdi} = n_{pdo} = 32$ for the 42 selected cores in Table 5.7.

5.8 Conclusion

We have proposed two versions of a NoC-compatible wrapper that requires minimal overhead on the test application time and area overhead. The previously proposed wrapper design did not handle the problem of inefficient bandwidth utilization. In this chapter, we have proposed two heuristics that find the best wrapper design for a given maximum bandwidth or maximum test application time, which is important for test schedule optimization.

The proposed wrapper does not incur large test time overhead (against the IEEE 1500 standard) for the same number of wrapper scan chains (about 3% for a very small circuit, and less than 0.25% for larger circuits). The wrappers scale well for large circuits. The advantage of the proposed wrapper is that NoC reuse is possible with only small test time overhead. With additional allowances on the area overhead, the proposed wrapper (Type 2) can efficiently utilize the NoC bandwidth with zero overhead on the test application time.

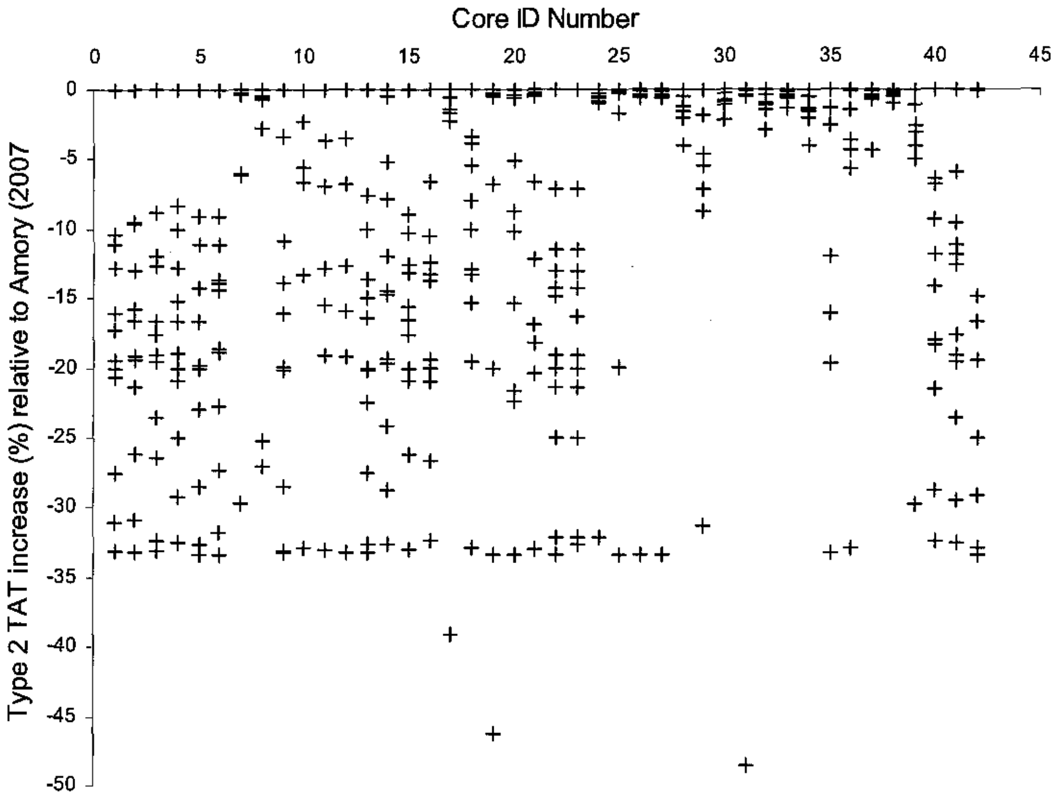


Figure 5.12: Test application time comparison comparison between the Type 2 wrapper and the wrapper in [96]. The wrapper configurations are determined for 16 unique values of maximum bandwidth, $B_{max} \in [1 \times f_m, 2 \times f_m, ..., 16 \times f_m]$ *bps* and $n_{pdi} = n_{pdo} = 32$ for the 42 selected cores in Table 5.7.

Compared to the NoC-compatible wrapper in [96], the enhanced Type 2 wrapper gives an average of 7.8% test time reduction for 42 selected SoC benchmark cores under various NoC bandwidth constraints.

Chapter 6

NoC Test Scheduling Through Bandwidth Sharing

6.1 Introduction

The NoC provides abundant plug-and-play capable communication resources, which makes the extraneous TAM overkill. As a result, the reuse of functional on-chip resources for test purposes is becoming more practical and more economical. Several research groups have published works on NoC test scheduling [61–63, 97, 102–105] utilizing the NoC as the delivery path for the transport of the test data from external testers to the CUT's. Test scheduling for the NoC router [63, 98] and crosstalk test of the interconnects [99] have also been discussed. In these approaches, each CUT is wrapped by an IEEE 1500 compatible wrapper in order to provide isolation and access during the test application.

The test scheduling methodologies proposed in [61–63, 97, 102–105] utilizes the NoC as test data transportation paths from external testers to the CUT's. In all these approaches, a dedicated path is established from the NoC input port to the CUT to transport the test vectors; another path is dedicated from the CUT to an output port for test response transportation. This is illustrated in Figure 6.1. One data-transfer path from Port 1 to Port 3 is used to test core A. Under the dedicated path approach, the whole bandwidth available along the selected data-transfer path is used. Therefore, another data-transfer path to test core B cannot be established between Port 2 and Port 3 because this new path

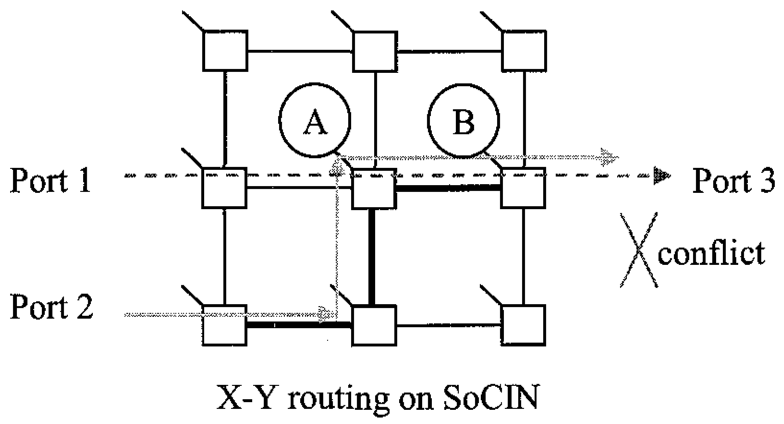


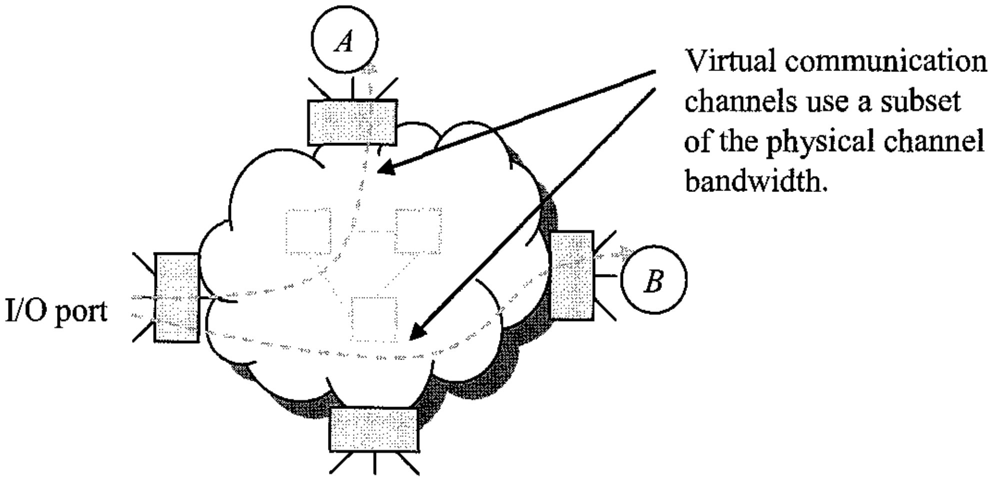
Figure 6.1: Dedicated path test methodology.

requires the resources currently being used by the first path to test core A.

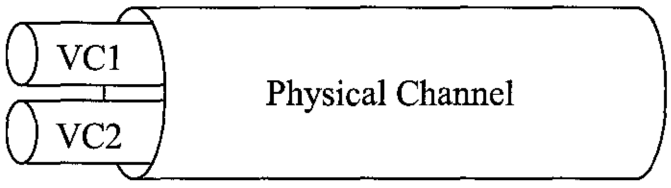
Dedicating a physical path to one core means that the path cannot be shared, thus preventing potential test concurrency—a useful tool for test schedule optimization. In addition, the dedicated path approach wastes the NoC bandwidth when the core test uses only a subset of the total physical channel bandwidth. A more efficient utilization of the large NoC bandwidth is to allow the large physical NoC channel bandwidth to be shared. This can be easily implemented because of the packet-switched network utilized by the NoC. Without making any changes to the NoC, the existing functional services provided by the NoC can be utilized during the test application time. In this chapter, our test scheduling method based on this concept is explained. To the best of our knowledge, we are the first to propose such method based on the bandwidth sharing scheme.

In addition, the methods proposed in [61–63, 97, 102–105] assumes that the test data will be delivered in a timely manner. This is difficult to justify because there is no guarantee provided by the packet-switching-based NoC, which uses multi-hop store-and-forward routers. Hence, the use of standard IEEE 1500 [3, 4] compatible wrapper cannot guarantee uncorrupted data loaded into the scan chains in every scan cycle.

To overcome this shortcoming, the authors in [7] propose a NoC wrapper which takes advantage of the guaranteed bandwidth and latency provided by the


A
B
Virtual communication channels use a subset of the physical channel bandwidth.
I/O port

(a) Bandwidth sharing through the network of routers.



(b) Two virtual channels inside a physical channel.

Figure 6.2: Conceptual representation of the bandwidth sharing scheme used in the proposed test scheduling methodology.

NoC to ensure test data integrity. While using the NoC as a TAM, the test data loading time of the NoC wrapper is comparable to the IEEE 1500 wrapper, which requires a more flexible but costly dedicated TAM, as implemented in [12, 21, 106]. However, the NoC wrapper requires much higher guaranteed bandwidth on the NoC than the actual rate of the test data loaded into the wrapper scan chains. This is further explained in Chapter 5 in which two complementary wrapper architectures are proposed in order to overcome the limitations of the NoC wrapper in [7].

In this chapter, we propose a test scheduling mechanism for NoC-based SoC, which utilizes the two types of complementary NoC wrappers for area cost and test application time (TAT) co-optimization [107, 108]. The proposed approach takes advantage of the NoC's ability to allocate a specific amount of sustained bandwidth for any particular packet-based connection called a *virtual channel*, making it possible to divide a physical connection for concurrent tests of multiple CUT's. The proposed bandwidth sharing achieves considerable reduction in test time, compared to the dedicated path approaches in [61–63, 97, 104, 105].

The rest of the chapter is organized as follows: Section 6.2 gives a brief overview of the proposed test scheduling scheme based on NoC bandwidth sharing. The NoC and IP core models are briefly described in Section 6.4. The test schedule and wrapper optimization methodology through bandwidth sharing is explained in Section 6.5. Some experimental results on selected benchmark circuits are given in Section 6.6. Finally, concluding remarks are offered in Section 6.7

6.2 Overview of the Proposed Bandwidth Sharing Scheme

The proposed test architecture utilizes the functional communication channel between a test source/sink and a circuit-under-test (CUT). Unlike the test strategies adopted in [62, 63, 97], the proposed approach does not restrict to any NoC network topology; it can be applied as long as minimum sustainable bandwidth and latency can be established and guaranteed during the test application of the target CUT, regardless of the NoC topology.

The test methodology in [62, 63, 97] treats the NoC interconnects similarly to a passive and unintelligent dedicated TAM. Test stimuli are transported to the CUT through an I/O port along a defined physical path through the NoC routers. The test responses are again transfered from the CUT back to the external tester through another path to a different I/O port, as illustrated by Figure 6.1. These I/O ports are artificially created by inserting multiplexers or bypasses between the chip's I/O pins and the NoC interfaces, or by establishing thru-paths (by means of thru-functions) through the embedded cores.

The proposed test strategy covered in this chapter uses the functional NoC interfaces and services as they are designed and meant to be functionally used. NoC interconnect provides various high-level and low-level services to enhance its efficiency as an advanced interconnect. Stimuli and responses are transported through existing functional I/O port(s), and uses a subset or all available NoC channel bandwidths to maximize efficiency. The proposed test methodology is primarily based on the utilization of bandwidth sharing. Bandwidths are reserved when virtual communication channels are established. A bandwidth-reserved virtual channel is similar to a fixed-width TAM, capable of transporting the specified amount of test stimuli and responses within the specified timing constraint.

The quality-of-service guarantees (i.e. bandwidth and latency guarantees) ensure that the test data are available at the CUT at the time it is required. Thus, a bandwidth-guaranteed virtual channel can be treated as a conventional TAM during the test scheduling process. In this chapter, the Æthereal [80] NoC, which implements data transfer through normal read/write transactions using the *shared-memory abstraction*, is used as an example in order to ease explanation.

6.3 Previous Works

This section explains some of the previous works on the test scheduling of core-based test through the NoC reuse.

For the NoC interconnect (i.e. router and network interface) testing, Aktouf [59] proposed a boundary-scan technique using the IEEE 1149.1 boundary-scan standard as the first phase of the integrated test strategy. The boundary-scan technique treats router test as a test of regular structures, by taking advantage of the regularity of the router design. A set of routers are groups into basic cell groups (GC's) as illustrated in Figure 6.3. External tester access is provided by the test access port (TAP) through a serial interface.

For the core test, Cota *et al.* [61, 97] and Liu *et al.* [104, 105] proposed dedicated path based approaches for a complete rectangular mesh network topology. All these works are based on the SoCIN network architecture that implements the XY-routing strategy. A short discussion of the SoCIN architecture is given in Section 4.4.1.

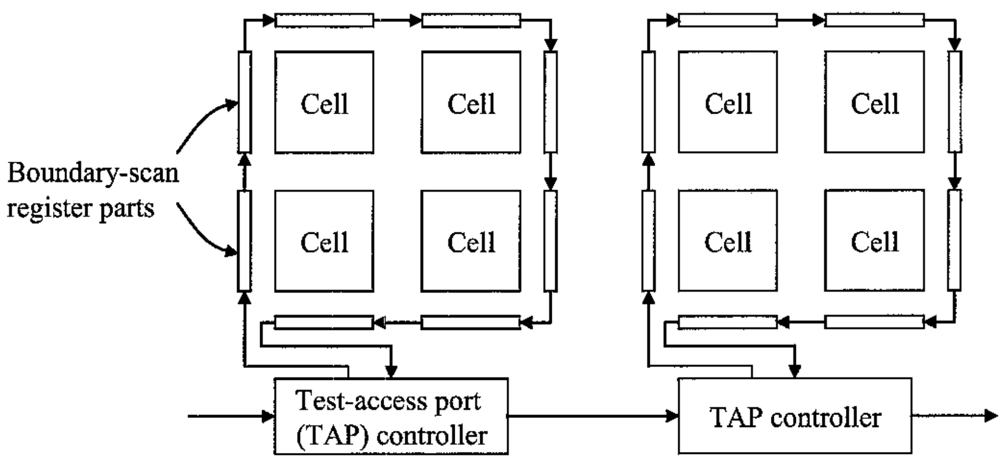
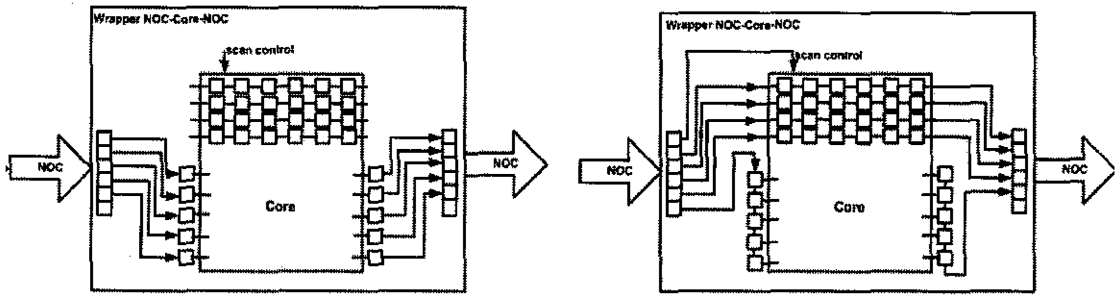

Boundary-scan register parts
Cell
Cell
Cell
Cell
Cell
Cell
Cell
Cell
Test-access port (TAP) controller
TAP controller

Figure 6.3: Basic-cell groups for router testing, accessible through the test-access port (TAP).

To enable network reuse, the test stimuli and responses are expressed as a set of packets to be transmitted through-out the network. In order to keep the wrapper as close as possible to the original design, the packets are defined so that each flit (a flow control unit) can be unpacked in one clock cycle. This requirement somewhat restricts the wrapper design. Each bit of each flit of a packet is to fill exactly one bit of the core scan chains. This is illustrated in Figure 6.4(b). The wrapper scan chains are designed in such a way to make sure that the flit size is enough to transport one bit of each scan chain. Therefore, the number of scan chains must not exceed the flit size. Similarly, if the number of wrapper scan chains is less than the flit size, some data bits carried by the flit (hence the packet) are ignored. The wasted bandwidth cannot be used for any other purpose because the path is dedicated to the test of the current core-under-test.

In terms of I/O utilization, Cota *et al.* and Liu *et al.* both require a pair of I/O port for the test of one core. One for the stimuli transportation, another for the responses transportation. Separate I/O ports are used for the stimuli and responses. Since the functional I/O ports are typically limited to only a few, depending on the functional requirements, artificial I/O ports are created. These artificial I/O's are illustrated in Figure 6.5, through cores 3, 4, 7, and 9.



(a) normal mode (b) test mode

Figure 6.4: Wrapper configurations for dedicated path approach.

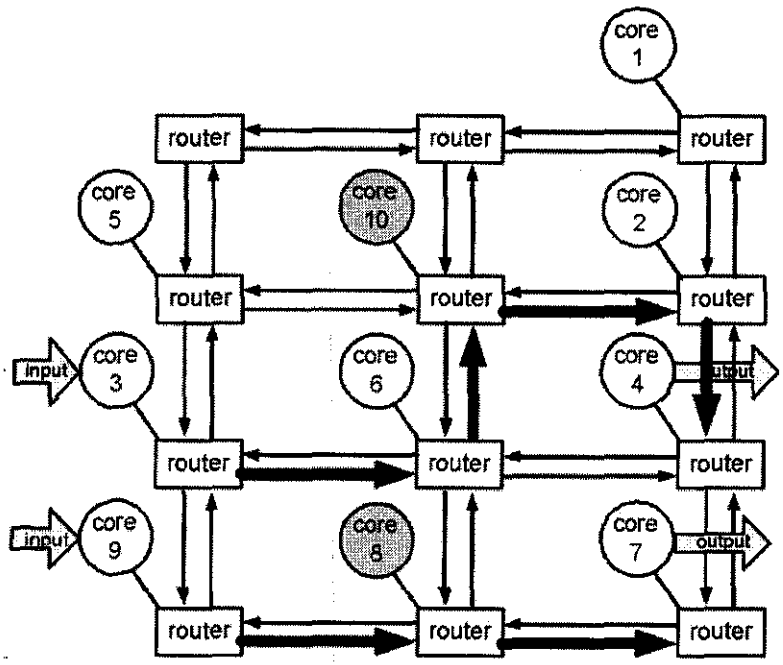


Figure 6.5: SoCIN-based NoC architecture for d695 [1]. The 2 × 2 artificial I/O ports are created by adding bypass multiplexers through cores 3, 4, 7, and 9.

The artificial I/O is created by inserting multiplexers into the IP cores to create bypasses. These bypasses provide direct interfaces from the primary I/O pins to the internal NoC routers.

The test scheduling process involves the tasks of efficiently allocating the free I/O ports and the router paths to the corresponding CUT's. All conflicts are resolved statically during the scheduling, therefore no network arbitration is required. The limitations of these approaches can be summarized as follows.

1. The scheduling methodology is applicable only to a square or rectangular mesh topology.
2. The allocation of a physical path to a particular CUT is wasteful and inefficient. This is because in some cases, the bandwidth required by the CUT is much less than the physical bandwidth available at each physical channel. For a wrapper scan chain that is narrower than the NoC channel, the remaining NoC data bits are ignored. This creates unnecessary underutilization of the NoC bandwidth, hence preventing the potential reduction in the test application time.
3. The creation of the artificial I/O port is intrusive and the addition of the multiplexers along a potentially-critical path could create a timing problem.

6.4 NoC and IP Core Models

Figure 6.6 shows the same SoC model described in Section 4.4.2 (Figure 4.10), but with the interface to the automatic test equipment (ATE) shown. Two virtual channels (VC) are shown connecting the ATE channel on port 1 to Core 1 and Core 2, respectively. Another VC connects the ATE channel on port 2 to Core 4. Each VC vc_k is guaranteed a minimum sustained bandwidth B_{vc_k}, where $\sum_k B^{i,j}_{vc_k} \leq B^{i,j}_{max}$. The term $B^{i,j}_{vc_k}$ and $B^{i,j}_{max}$ represent the reserved bandwidth for vc_k and the maximum link bandwidth, respectively, between each pair of routers R_i and R_j along the vc_k path. If $B^{i,j}_{vc} < B^{i,j}_{max}$ for some link $R_i \rightarrow R_j$, the unreserved bandwidth can be allocated to other VC's in order to allow simultaneous test applications of multiple CUT's.

Intellectual property (IP) core inputs and outputs (I/Os) shown in Figure 6.7 consist of primary inputs (PI), primary outputs (PO), scan inputs (SI) and scan outputs (SO). A subset of the PI's can be categorized into *primary data inputs* (PDI) and *primary control inputs* (PCI), which are connected to the NoC input port. Correspondingly, on the output side, there are *primary data outputs* (PDO)

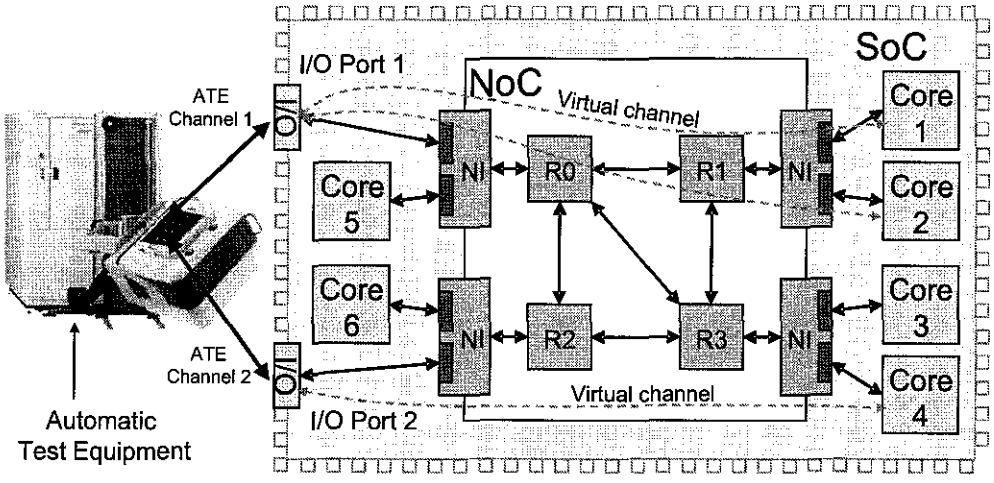


Figure 6.6: SoC model based on the Æthereal NoC.

and *primary control outputs* (PCO). The PDI's and PDO's are used to carry the test vectors from the ATE to the CUT, and the test responses from the CUT to the ATE, respectively. The remaining PI/PO's (PI' and PO') are connected to other parts of the SoC, which includes other cores, and the SoC's primary I/O's.

The test scheduling described in this chapter refers to these NoC and IP core models. The NoC architecture and the guaranteed services are the basis of the proposed strategy. In addition, the IP core interfaces and design-for-testability (DfT) infrastructure are used by the proposed NoC-compatible wrappers described in Chapter 5.

6.5 NoC Test Scheduling

Unlike the test scheduling scheme based on the dedicated path approach proposed by all the previously published literature, we utilize a more efficient method of sharing the available interconnect bandwidth. This method proves to be much more efficient. In addition, it is more compatible with the NoC's intended functionality, which provides as a service the means of allocating the interconnect bandwidth.

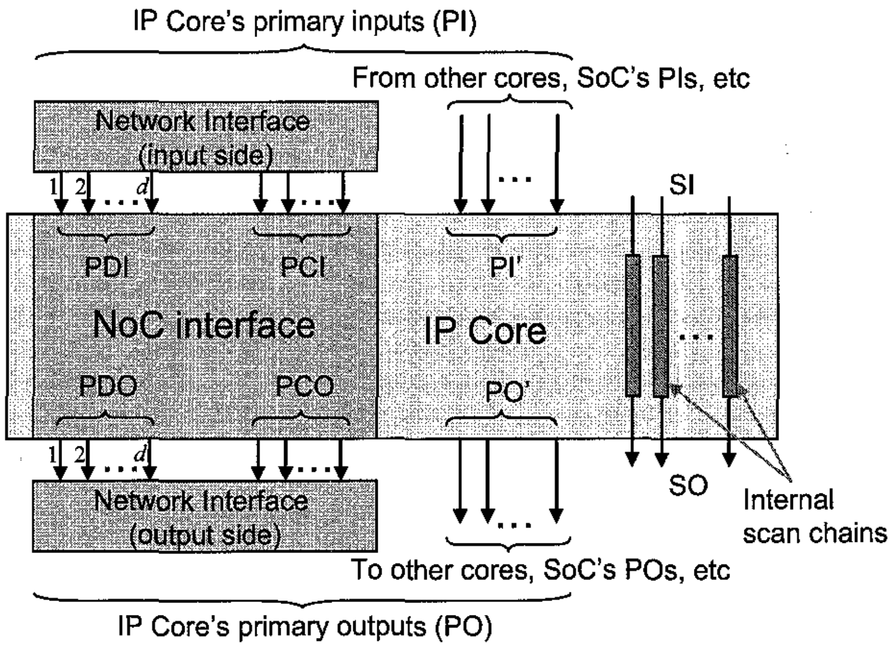

IP Core's primary inputs (PI)
From other cores, SoC's PIs, etc
Network Interface (input side)
1 2 ... d
SI
PDI
PCI
PI'
NoC interface
IP Core
PDO
PCO
PO'
1 2 ... d
SO
Internal scan chains
Network Interface (output side)
To other cores, SoC's POs, etc
IP Core's primary outputs (PO)

Figure 6.7: IP core model interfaced to the NI port.

6.5.1 Characteristics of NoC-compatible Wrappers

The test scheduling described in this section uses the Type 1 (Figure 5.4) and type 2 (Figure 5.7) wrappers proposed in Chapter 5. As explained in Chapter 5, the two proposed NoC-compatible wrappers are complementary to each other. The wrapper characteristics are summarized in Table 6.5.1. Type 1 wrapper requires one wrapper boundary register (WBR) cell for each I/O pin. The Type 2 wrapper uses two WBR cells for each data-carrying input and output pins in addition to two additional WBR cells for each wrapper scan chain. All these additional WBR cells are used to provide a flexible bandwidth matching architecture. As a result, Type 2 wrapper is always efficient in terms of NoC bandwidth utilization.

The Type 1 wrapper is a direct extension of the standard IEEE 1500 wrapper, but with the added capability to accept the test data from the data input and to match the fixed bit width data to any arbitrary number of wrapper scan chains. In addition to being able to take the functional data, the Type 1 wrapper also provides bit width matching between the data port and the scan port. The only

Table 6.1: Complementary characteristics of Type 1 and Type 2 NoC-compatible wrappers proposed in Chapter 5

	NoC-compatible wrappers	
Characteristics	Type 1	Type 2
Area cost	Low	High
Bandwidth utilization	Inefficient	Efficient

limitation of the Type 1 wrapper is when the data bit width and the number of wrapper scan chains are not compatible, as explained in Section 5.5.1.

Under such unsuitable configuration, the Type 1 wrapper sacrifices bandwidth efficiency in order to interface the NoC with the wrapper scan chains. This is achieved by ignoring one or more data-carrying bits so that the remaining number of data bits is compatible with the number of wrapper scan chains.

The Type 2 NoC wrapper in Figure 5.7 is designed to complement the Type 1 wrapper in this aspect. The load/shift registers translate the PDI bit-width, n_{pdi}, into the number of wrapper scan chains, n_{sc}, using parallel-serial shift registers shown in Figures 2.8(a) and 5.3(a). As a result, the required NoC bandwidth matches the scan bandwidth. The TAT for the Type 2 NoC wrapper is also the same as the IEEE 1500 wrapper. This is achieved at the cost of a larger area overhead and a more complex control scheme to realize the bit-width conversion.

The two types of wrappers, with rather opposite characteristics, can be used effectively to prevent unnecessary increase in the test application time of an individual core while also optimizing the area overhead.

6.5.2 Test Scheduling through NoC Bandwidth Sharing

NoC is designed as an advanced SoC interconnect [75–90] to provide a high bandwidth and modular infrastructure for on-chip communications. As such, in a typical SoC implementation the internal NoC bandwidth is typically larger than the external I/O bandwidth.

Definition 6.1 *Internal NoC bandwidth* is the router-to-router and router-to-embedded cores link bandwidth or capacity (in bits-per-second) inside the NoC architecture as shown in Figure 6.8.

Definition 6.2 *External I/O bandwidth* is defined as the link bandwidth or capacity from an I/O interface unit to the external devices.

In Figure 6.8, router-to-router bidirectional links are rated at 16 Gbps (i.e. 32-bit wires at 500 MHz for each direction). The external interface through the I/O port is rated half the internal bandwidth at 8 Gbps. Each core is labeled with the corresponding scan rate. For example, core C_1 has 16 wrapper scan chains. When tested at the scan frequency of 100 MHz, it requires the test data at the rate of 16 bits $\times$ 100 MHz, or 1.6 Gbps (Giga bits per second).

The test of core C_1 utilizes only a subset of the bandwidth on the I/O port, and between routers $R1$ and $R2$. With the bandwidth sharing approach, we can allow multiple cores to be tested concurrently. For example, simultaneous testing of C_1, C_3, and C_6 requires 8 Gbps on the I/O port, 3.2 Gbps on $R1-R3$ link, 4.8 Gbps on $R1-R2$ link, and 3.2 Gbps on $R2-R4$ link. The shared I/O bandwidth limits further test concurrency. Nevertheless, bandwidth sharing approach allows more efficient use of NoC bandwidth compared to the dedicated path approaches.

The proposed approach is applicable to any NoC architecture that implements the bandwidth reservation scheme, such as the time-domain multiplexing (TDM) scheme implemented by Æthereal. The NoC routers are interfaced to the cores through a buffer-based network interface (NI) architecture, as shown in Figure 6.9. Test application can be implemented between the Master (Automatic Test Equipment) and the Slave (Core Under Test).

Because of the guaranteed bandwidth, the incoming data buffer at the core is always non-empty. At the core wrapper (Figure 5.4 and 5.7), new data availability is signaled by the pci[0] and pci[1] control signals; depending on the write transaction protocol, the signals could be DATA_STROBE, DATA_VALID, etc. as used by the corresponding handshake protocol. These handshake signals are detected by the wrapper *Controller* (Figures 5.4 and 5.7), which then generates the necessary sequence of control signals for parallel-serial conversion at the wrapper's inputs and outputs. After the data from the PDI port is shifted into

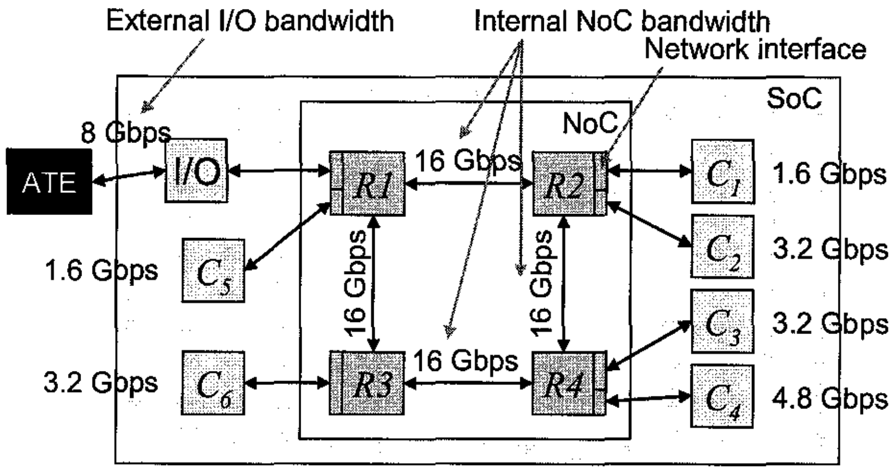


Figure 6.8: Illustrative example of the NoC-based SoC model used by the proposed bandwidth sharing approach.

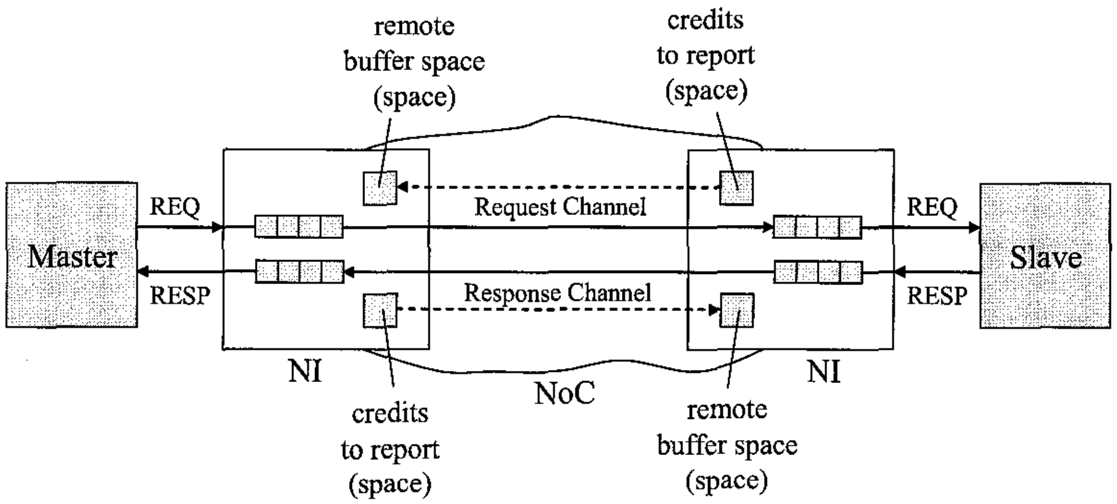


Figure 6.9: Buffer-based virtual channels (request and response) between a master and a slave. Illustration by Radulescu *et al.* 2005 [80].

the wrapper scan chains, an acknowledgment signal is generated to enable the network interface (NI in Figure 6.9) to deliver the subsequent data.

This buffer-based architecture with credit-based flow control transforms the bursty packet-switched data in the NoC into a steady stream of data between the Master and the Slave. The buffer architecture also separates the NoC's clock from the Core's clock; therefore, the cores' clocks can be independent from each other. For that reason, the proposed approach can also be applied to multi-clock SoC's.

In this chapter, we consider the test application of such SoC's utilizing the external tester as the test source and sink. The ATE ports are connected to the SoC through these low bandwidth I/O ports, as illustrated in Figure 6.6 and Figure 6.8. The test data are transferred into the chip through the functional write transactions. We will assume that a virtual channel can always be established from the I/O port to the target CUT as long as $\sum \{virtual\ channel\ bandwidth\} \leq \{I/O\ bandwidth\} \leq \{internal\ NoC\ bandwidth\}$. Under this assumption, the wrapper area and test time co-optimization problem addressed in this chapter can be formulated as an I/O bandwidth distribution and core test scheduling problem as follows:

Problem 6.1 $[\Psi_S]$ Given an SoC C with M cores, a maximum I/O bandwidth, $B^{i/o}_{max}$ *bps*, and a scan frequency for all cores, f_m, where each core consists of n_{ip} functional inputs, n_{op} functional outputs, n_{bi} bidirectionals, k internal scan chains of length $l_1, l_2, ..., l_k$, for each core $c_i \in C$ determine

(1) the wrapper type and the allocated I/O bandwidth, $B_{scheduled}[c_i]$, for the test data transportation, and
(2) the starting time, $t_{start}[c_i]$, and end time, $t_{end}[c_i]$, of the test application

such that the total test application time and the area overhead are optimized under given priority weights α and β, respectively, where $\{\alpha, \beta\} \in [0,1]$ and $\alpha + \beta = 1$. ∎

Before explaining the schedule optimization algorithm (Section 6.5.6), we first clarify two required components of the algorithm in sections 6.5.3 and 6.5.5.

6.5.3 Optimum Wrapper under Bandwidth Constraint

In order to achieve the objective (1) of Problem 6.1 (Ψ_S), we defined in Section 5.6 the problems of optimizing the number of wrapper scan chains (n_{sc}) for both the Type 1 and the Type 2 wrappers under given constraints. Problem 5.1 (Ψ_B) on page 144 was defined as a wrapper design and optimization problem under a given maximum bandwidth constraint, B_{max}. A similar problem (Problem 5.2 (Ψ_T) on page 145) was defined but instead of NoC bandwidth, a given maximum test time, T_{max}, is considered as constraint.

It was shown in [12], and illustrated in Figure 5.10 (top diagram), that the TAT of a core is a monotonic decreasing function with respect to increasing number of wrapper scan chains. Therefore, the optimum solution to Ψ_B can be found in polynomial time, even when an exhaustive search is used. In Section 5.6 we implemented a binary search function to find the optimum wrapper configuration. The wrapper configuration is said to be optimum under the given maximum bandwidth when all objectives of Problem 5.1 (Ψ_B) are met.

The optimum wrapper ensures minimum test application time (objective (*i*)), without exceeding the allocated maximum bandwidth (objective (*ii*)). The third objective is only effective when there are multiple solutions that meet the first two objectives. Such scenarios are possible because of the staircase properties of wrapper's test time with respect to the number of wrapper scan chains, as shown in Figure 5.10 (top diagram). The third objective aims to select the wrapper with the smallest number of scan chains, without violating the first objective. Such point is called a Pareto-optimal point (the concept of Pareto-optimal was discussed in [106]). A similar search algorithm was also implemented for problem Ψ_T, for a given maximum test application time.

6.5.4 Cost Function for Wrapper Optimization

The process of wrapper optimization involves choosing the wrapper type and the corresponding number of wrapper scan chains under certain constraints. As discussed in Section 5.6 and illustrated by Figure 5.9, the decision on the wrapper type and configuration is made during the scheduling process. The decision is made based on the "available" time period or bandwidth. The available values are illustrated by T_1, T_2, B_1, and B_2 in Figure 5.9. Based on the available information (test time or bandwidth), we can determine the following three variables for the Type 1 and Type 2 wrappers. These three variables are used to make the cost function in the proposed heuristic algorithm when choosing the wrapper type.

- the area cost (overhead) of the wrapper,
- the NoC bandwidth required by the wrapper, and
- the core test time effected by the wrapper.

The area cost/overhead of the wrappers is contributed mainly by the quantity of the boundary cells. We will assume that the area overhead due to the wrapper controller is comparable for both wrappers, therefore will not be used when deciding the wrapper type. The area overhead for Type 1 and Type 2 wrappers can be estimated by equations (6.1) and (6.2), respectively. The extra $(+n_{pdi} + n_{pdo} + 2 \cdot n_{sc})$ in equation (6.2) are due to the additional input/output buffers in the Type 2 wrapper (Figures 5.4 and 5.7) that perform bit-width matching. Equation (6.3) gives the total extra cost of using a Type 2 instead of the Type 1 wrapper. Equation (6.4) gives the opposite cost.

$$H_{t1} = n_{ip} + n_{bi} + n_{op} \tag{6.1}$$

$$H_{t2} = n_{ip} + n_{bi} + n_{op} + n_{pdi} + n_{pdo} + 2 \cdot n_{sc} \tag{6.2}$$

$$Cost_{(t1 \to t2)} = \alpha \cdot \left(\frac{T_{t2} - T_{t1}}{T_{t1}} + \frac{B_{t2} - B_{t1}}{B_{t1}} \right) + \beta \cdot \frac{H_{t2} - H_{t1}}{H_{t1}} \tag{6.3}$$

$$Cost_{(t2 \to t1)} = \alpha \cdot \left(\frac{T_{t1} - T_{t2}}{T_{t2}} + \frac{B_{t1} - B_{t2}}{B_{t2}} \right) + \beta \cdot \frac{H_{t1} - H_{t2}}{H_{t2}} \tag{6.4}$$

For a given maximum available bandwidth, B_{max}, the optimum configuration of a core c_i is determined by solving $\Psi_B(c_i, B_{max})$ to obtain the respective core test time effected by the wrapper (T_{t1} and T_{t2}) and required bandwidth (B_{t1} and B_{t2}) for the Type 1 and the Type 2 wrappers, respectively. Similarly, given a maximum available test time, T_{max}, all four variables can be calculated using $\Psi_T(c_i, T_{max})$.

If $Cost_{(t1 \to t2)} < Cost_{(t2 \to t1)}$, then the Type 2 wrapper is selected as a better wrapper configuration for the given B_{max} or T_{max}. Otherwise, the Type 1 wrapper is chosen. This cost function will be the basis for wrapper selection under given cost weights α and β, as defined in Ψ_S. The weighting variables give us the option of whether to prioritize the area cost or the test time during the optimization.

The weight α is defined for the test application time in equations (6.3) and (6.4). However, α is shared by the test application time and the required bandwidth because of their inverse relationship. Test application time can be minimized by giving the maximum possible bandwidth, or it can be maximized by giving minimum bandwidth to the NoC-compatible wrappers. All this can be done without any effect on the area overhead. In order to prevent favoring either test time over area overhead or vice-versa (i.e. by varying the bandwidth), we consider the normalized sum of test time and bandwidth in the equations (6.3) and (6.4) because of their inverse relationship.

6.5.5 Lower Bound on Test Time

The authors in [21] proposed an architecture independent tight lower bound for dedicated TAM based test application, considering both fixed and flexible length internal scan chains. In this section, a similar lower bound based on bandwidth utilization is explained for use in the optimization algorithm in Section 6.5.6. The first lower bound is based on the *dominant core* effect. For each core $c_i \in C$, assuming that it is given the maximum available bandwidth, $B^{i/o}_{max}$, its test time can be determined by $T(\Psi_B(c_i, B^{i/o}_{max}))$, which represents the TAT returned by the Ψ_B search algorithm for Core c_i when the given maximum bandwidth is $B^{i/o}_{max}$. Even with unlimited bandwidth, the TAT of an SoC C cannot be shorter than the TAT of the dominant core $c_i \in C$. Therefore the first lower bound can be summarized as

$$T^1_{LB} = max_{i \in C}\{T(\Psi_B(c_i, B^{i/o}_{max}))\} \tag{6.5}$$

For a bounded I/O bandwidth $B^{i/o}_{max}$, T^1_{LB} does not represent a meaningful lower bound. Therefore, a tighter lower bound based on the I/O capacity to transfer test vectors into the SoC is formulated as follows. This limited bandwidth assumption means that the overall test time is at least equal to or greater than the dominant core. Assuming that the wrapper for a core c_i forms one scan chain, its TAT can be represented by equation (6.6) where scan-in depth, $si =$

$n_{ip}+n_{bi}+\sum_k l_k$, scan-out depth, $so = n_{op}+n_{bi}+\sum_k l_k$, and v_c is the number of test vectors. The second lower bound can be calculated as in equation (6.7), where f_c is the scan frequency for all cores. The overall lower bound is the maximum of T^1_{LB} and T^2_{LB} (equation (6.8)).

$$T(c_i) = (max(si, so) + 1) \times v_c + min(si, so) \tag{6.6}$$

$$T^2_{LB} = \frac{\sum_{c_i \in C} t(c_i)}{B^{i/o}_{max}/f_c} \tag{6.7}$$

$$T_{LB} = max(T^1_{LB}, T^2_{LB}) \tag{6.8}$$

Equation (6.7) is effective for the case when the test application time is constrained by the maximum I/O bandwidth of the NoC, $B^{i/o}_{max}$. In this case, increasing the scan frequency, f_c, will not increase the test application time as might be indirectly suggested by equation (6.7). Increasing the test frequency will not by itself decrease the test application time because the test data cannot be transported quickly enough. In order to ensure that the test data is available at the scan inputs at every scan clock, when the scan frequency is increased we need to compensate by reducing the number of wrapper scan chains so that the required bandwidth is constant. This is shown in the following equation, where B = bandwidth, n_{sc} = number of wrapper scan chains, and f_c = scan frequency.

$$B = n_{sc} \times f_c \tag{6.9}$$

In equation (6.6), $T(c_i)$ represents the test application time in terms of the number of scan and capture cycles for a given number of wrapper of scan chains (one scan chain in this case). Equation (6.7) gives the overall test application time (in number of scan and capture cycles) for the SoC under a given maximum bandwidth, $B^{i/o}_{max}$. If the scan frequency is increased, equation (6.7) indicates that the number of scan and capture cycles increase (because of the required reduction in the number of wrapper scan chains), but does not necessarily increase the time (in seconds) because of the increase of scan frequency.

If we rewrite equation (6.7) in terms of time (seconds) instead of the number of scan clock cycles, the term f_c would cancel, resulting in an equation that is independent of the scan frequency. This is true for the condition under which the equation is formulated, i.e. the test time is constrained by the maximum I/O bandwidth.

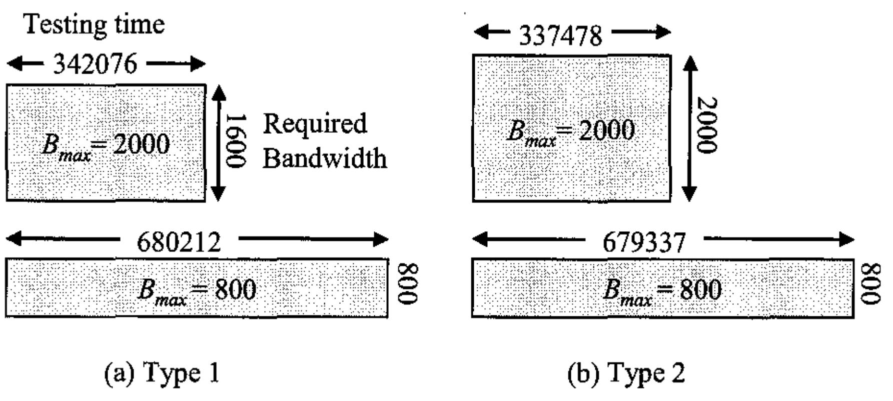


Figure 6.10: Rectangles represent tests of Core 6 of p93791 [1] benchmark circuit.

6.5.6 Schedule Optimization through Rectangle Packing

We now introduce the concept of rectangles to represent core tests, then explain a flexible scheduling methodology based on NoC bandwidth sharing, which is inspired by the scheduling algorithm in [106]. The use of rectangles have previously been proposed in [42, 106] for dedicated TAM based scheduling approach. In this chapter, the height of a rectangle represents the required NoC bandwidth to obtain the test application time represented by the horizontal length. Figure 6.10 illustrates two pairs of rectangles, each representing the test of Core 6 of p93791 circuit (ITC'02 benchmark [1]) when B_{max} = 2000 Mbps and 800 Mbps, respectively. For this example, the NoC port's PDI/PDO bit-width is $n_{pdi} = n_{pdo} = 64$ bits.

The top left rectangle is obtained using the wrapper optimization algorithm Ψ_B described in Section 6.5.3, when given as input the maximum allocated bandwidth, $B^{vc}_{max} = 2,000$ Mbps. The algorithm iteratively searches for the wrapper configuration that produces the smallest test application time, which fulfills the Pareto-optimal criteria, under the bandwidth constraint. Since the Type 1 wrapper cannot effectively utilize all the allocated bandwidth, the algorithm finds the next Pareto-optimal point with a TAT of $342,076$ clock cycles which requires $1,600$-Mbps NoC bandwidth. The same procedure is repeated for the Type 2 wrapper. With a more efficient bandwidth matching architecture, the Pareto-optimal wrapper is found with a TAT of 337,478 clock cycles and a required bandwidth of 2,000 Mbps (top right rectangle). For $B^{vc}_{max} = 2,000$ Mbps, these two wrapper configurations are candidates for scheduling.

The complete scheduling algorithm is given in Algorithm 6.1. It starts by obtaining the *preferred bandwidth* for each core in the SoC C. As illustrated in Figure 6.11, the preferred bandwidth results after configuring the core wrapper with the number of scan chains in the "high gain" region. Gain represents the potential reduction in TAT of a core per additional unit of bandwidth allocated to that core. Therefore, rather than allocating more bandwidth to a core when it is already in the low gain region, it would be wiser to assign that bandwidth to a different core that is still in the high gain region.

Algorithm 6.1 OptimizeSchedule $(C, B^{i/o}_{max}, v_{gain}, v_{bottleneck})$
Data Structure: *Schedule*

```
    $t_{start}[c_i]$; /*start time of Core $c_i$*/
    $t_{end}[c_i]$; /*end time of Core $c_i$*/
    $B_{scheduled}[c_i]$; /*allocated bandwidth for Core $c_i$*/
```

```
 1. PreferredBandwidth $(C, B^{i/o}_{max}, v_{gain}, v_{bottleneck})$
 2. $B_{free} \leftarrow B^{i/o}_{max}$; $t_{current} \leftarrow 0$;
 3. While $C \neq \emptyset${
 4.    If $B_{free} > 0${
 5.       If Core $c_i \in C$ can be found such that
              $B_{pref}[c_i] \leq B_{free}$ AND $T_{pref}[c_i]$ is maximum {
 6.          If $(C - \{c_i\} = \emptyset)$
 7.             ScheduleLastCore $(c_i)$;
 8.          Else
 9.             UpdateSchedule $(c_i, B_{pref}[c_i])$;}
10.       Else {
11.          Find $t_{next} \leftarrow t_{end}[c_i]$ such that $t_{end}[c_i] > t_{current}$ AND $t_{end}[c_i]$ is mini-
   mum;
12.         If Core $c_i$ can be found such that $(t_{current} + T(\Psi_B(c_i, B_{free})) \leq t_{next}$
                AND $B(\Psi_B(c_i, B_{free}))$ is maximum) {
13.           UpdateSchedule $(c_i, B_{free})$;}
14.         Else {
15.           DistributeFreeBandwidth ();
16.           $B_{idle} \leftarrow B_{free}$;
17.           $B_{free} \leftarrow 0$;}}}
18.     Else {
19.       Find $t_{next}$ as in Line 11;
20.       $t_{current} \leftarrow t_{next}$;
21.       $B_{free} \leftarrow B_{idle}$;
22.       For every Core $c_i$ such that $t_{end}[c_i] = t_{current}$ {
23.         $B_{free} \leftarrow B_{free} + B_{scheduled}[c_i]$;}}}
24. OptimizeMaxEndTime (*Schedule*);
25. Return *Schedule*;
```

Algorithm 6.2 describes the algorithm to determine the preferred bandwidth for all cores. In line 28, a proper value of input percent v_{gain} shifts the target TAT from $T_{max-pareto}$ to the high gain region. Figure 6.11 illustrates some of the variables, and show how $T_{target1}$ is calculated using the variable v_{gain}. Lines 26-29 are evaluated for both Type 1 and Type 2 wrapper configurations. For every core $c_i \in C$, equations (6.1)-(6.4) are evaluated to determine the best wrapper type, for which the value of $T_{pref}[c_i]$ is returned. The same wrapper selection procedure is performed at line 12 when evaluating $T(\Psi_B(c_i, B_{free}))$.

In some cases where the test application time is dominated by a large core such as Core 6 of p93791, selecting the high gain region for Core 6 could potentially make it a bottleneck core, thus preventing further reduction of TAT. In order to handle this kind of special cases, we need to be able to allocate as much bandwidth as possible to these potential bottleneck cores. In line 33, the variable $v_{bottleneck}$ together with the lower bound, T_{LB} (equation (6.8)), ensures that bottleneck cores are allocated larger preferred bandwidth, even if it is in the low gain region.

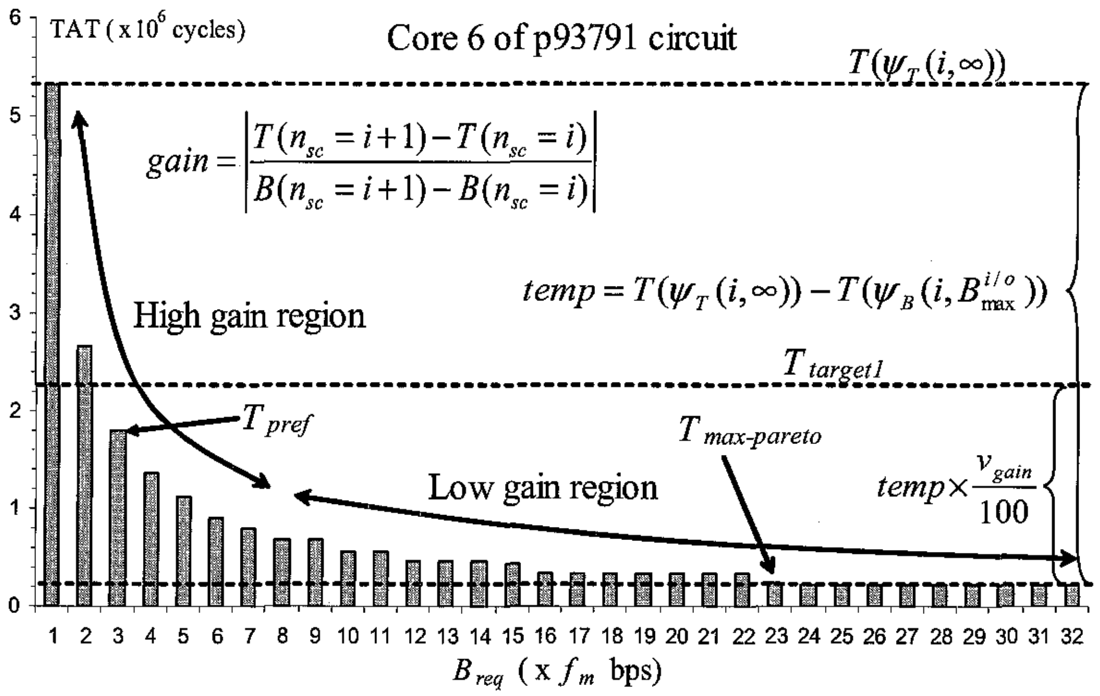


Figure 6.11: High (preferred) and low gain regions. $T_{pref} = \Psi_T(c_i, T_{target1})$; line 29 of Algorithm 6.2.

Algorithm 6.2 PreferredBandwidth $(C, B^{i/o}_{max}, v_{gain}, v_{bottleneck})$

```
26. For each Core $c_i \in C${
27.     $temp \leftarrow T(\Psi_T(c_i,\infty)) - T(\Psi_B(c_i, B^{i/o}_{max}))$;
28.     $T_{target1} \leftarrow T(\Psi_B(c_i, B^{i/o}_{max})) + (temp \times v_{gain}/100)$;
29.     $T_{pref}[c_i] \leftarrow T(\Psi_T(c_i, T_{target1}))$;}
30. $avgT_{pref} \leftarrow$ Average $T_{pref}[c_i]$ for all $c_i \in C$;
31. $T_{LB} \leftarrow max(T^1_{LB}, T^2_{LB})$; /*lower bound, equation (6.8)*/
32. For each Core $c_i \in C${
33.     $T_{target2} \leftarrow min\{v_{bottleneck} \times avgT_{pref}, T_{LB}, T_{pref}[c_i]\}$;
34.     $B_{pref}[c_i] \leftarrow B(\Psi_T(c_i, T_{target2}))$;
35.     $T_{pref}[c_i] \leftarrow T(\Psi_T(c_i, T_{target2}))$;}
```

The process begins with setting the current time, $t_{current} = 0$. During the scheduling process, a core is assigned its preferred bandwidth, B_{pref}, if the currently unused bandwidth, B_{free}, at the current time, $t_{current}$, is more than or equal to B_{pref} (Line 9). Otherwise, the core $c_i \in C$ that leaves minimum B_{free} after it is scheduled, is assigned bandwidth of $B_{req} = B(\Psi_B(c_i, B_{free})) \leq B_{free}$, that can be effectively utilized by the core c_i (Lines 12-13). $T(\Psi_B(c_i, B_{free}))$ on the other hand, returns the corresponding TAT. Scheduling a new core c_i involves assigning several variables—$t_{start}[c_i]$, $t_{end}[c_i]$, and $B_{scheduled}[c_i]$—and updating B_{free} and the list of unscheduled cores, C (Algorithm 6.3).

Algorithm 6.3 UpdateSchedule (c_i, B_{given})

```
36. $t_{start}[c_i] \leftarrow t_{current}$;
37. $t_{end}[c_i] \leftarrow t_{current} + T(\Psi_B(c_i, B_{given}))$;
38. $B_{scheduled}[c_i] \leftarrow B(\Psi_B(c_i, B_{given}))$;
39. $B_{free} \leftarrow B_{free} - B_{scheduled}[c_i]$;
40. $C \leftarrow C - \{c_i\}$;
```

When no more cores can be scheduled at $t_{current}$ while $B_{free} > 0$ (Lines 14-17), the core c_i, whose $t_{start}[c_i] = t_{current}$ and $t_{end}[c_i]$ is maximum, is allocated the remaining unused bandwidth. This is repeated until either no more $t_{end}[c_i]$ reduction of such cores is possible or $B_{free} = 0$. At lines 18-23, the current time and available bandwidth are updated before the *while* loop is reevaluated.

When scheduling the last core (Line 7), the core start time and assigned bandwidth is chosen such that t_{end} is minimum. This is illustrated in Figure 6.12(a) where three possible options are shown by the dotted rectangles. After all the cores are scheduled, in the final step (line 24), the current schedule of core c_i whose $t_{end}[c_i]$ is maximum, is reconsidered for further optimization. Without modifying the schedule for other cores, core c_i is rescheduled such that the new $t_{end}[c_i]$ is

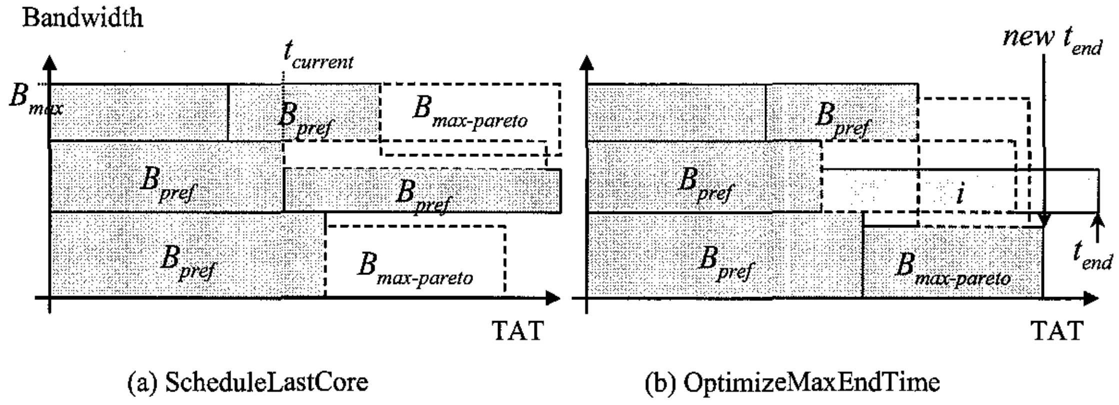


Figure 6.12: Further optimizing the schedule. Dotted rectangles represent possible schedule/wrapper configurations.

minimum (Figure 6.12(b)). This process is repeated until no more reductions can be made to t_{end}.

6.6 Experimental Results

In this section, we present experimental results for several modified ITC'02 benchmark [1] circuits (d695noc, p93791noc, p22810noc). The wrappers in Figures 5.4 and 5.7 utilize the PDI/PDO interface between the core and the NI in its operation. From the design perspective, the cores whose $n_{ip} + n_{bi} < n_{pdi}$ or $n_{op} + n_{bi} < n_{pdo}$ cannot be functionally interfaced to the NoC. As a result, two, four, and five small cores are excluded from each of the benchmark circuits when $n_{pdi} = n_{pdo} = 32$. In addition, the optimum values (determined iteratively) of $v_{gain} \in [0..9]$ and $v_{bottleneck} \in [1..5]$ are used, with the scan frequency, $f_m = 100\ MHz$. The TAT reported in this chapter is in number of scan clock cycles, where each cycle is equivalent to $1/f_m$ or $0.01\mu s$. The computation time is less than 10 seconds for the largest circuit.

Table 6.2: Area overhead comparison between Type 1 and Type 2 wrappers.

Benchmark Circuit Characteristics					DFT (NoC-compatible wrappers)					
					Type 1		Type 2 (32-bit PDI)		Type 2 (64-bit PDI)	
Name	#Cores	#I/Os	#SFF	#NOT gates	#NOT gates	Hardware overhead	#NOT gates	Type 2/ Type 1	#NOT gates	Type 2/ Type 1
d695noc	10	1,845	6,384	153,216	57,195	37.3%	67,135	6.5%	77,055	13.0%
p22810noc	*28*	4,283	24,723	593,352	132,773	22.4%	160,605	4.7%	188,381	9.4%
p93791noc	*32*	6,943	89,973	2,159,352	215,233	10.0%	247,041	1.5%	278,785	2.9%

Table 6.3: TAT for several hardware cost (β) and time cost (α) weights.

Cost weights		p93791noc $B_{max}^{i/o} = 6400$ Mbps ($T_{LB} = 435{,}039$)			p22810noc $B_{max}^{i/o} = 6400$ Mbps ($T_{LB} = 102{,}965$)			d695noc $B_{max}^{i/o} = 3200$ Mbps ($T_{LB} = 16{,}701$)		
β	α	AOH	TAT	$\%/T_{LB}$	AOH	TAT	$\%/T_{LB}$	AOH	TAT	$\%/T_{LB}$
0.00	1.00	9,303	464,252	6.7	5,768	122,091	18.6	2,396	17,827	6.7
0.25	0.75	8,653	464,252	6.7	5,680	122,280	18.8	2,300	17,827	6.7
0.50	0.50	7,673	471,175	8.3	5,698	122,280	18.8	2,300	17,827	6.7
0.75	0.25	7,673	471,175	8.3	5,412	130,591	26.8	2,110	18,184	8.9
1.00	0.00	6,557	483,411	11.1	3,810	134,466	30.6	1,676	18,494	10.7

Table 6.2 tabulates the comparison of Design-For-Testability (DFT) costs between Type 1 and Type 2 wrappers and the SoC benchmark circuits. The circuit size for the ITC'02 benchmark circuits are not given, therefore we estimate the circuit size in terms of the equivalent number of NOT gates for the given number of scan flip-flops (SFF). Each scan cell and wrapper cell is estimated to be equivalent to 24 NOT gates and 31 NOT gates, respectively.

Column labeled Type 1 gives the percent overhead of the Type 1 NoC wrapper (calculated using equation (6.1)) over the SoC circuit. For the largest circuit in the ITC'02 benchmark, the overhead is 10%. For the smaller circuit (d695), the overhead is as high as 37.3%. The Type 1 wrapper cell overhead is the same as the standard IEEE 1500 wrapper overhead.

When we consider the additional area overhead of a Type 2 wrapper (on top of the overhead of Type 1 wrapper), the value ranges between 1.5% to 6.5% (for 32-bit PDI/PDO) and between 2.9% and 13% (for 64-bit PDI/PDO), for the selected circuits. The additional hardware overhead of the Type 2 wrapper is not insignificant; therefore the proposed optimization method is necessary. The calculation is based on a single wrapper scan chain (i.e. $n_{sc} = 1$). The Type 2 hardware overhead would increase slightly for larger number of wrapper scan chains as indicated by equation (6.2).

In Table 6.3, the weights of area overhead cost (β) and TAT cost (α) are varied according to the constraints defined in Ψ_S. In Table 6.3 and Table 6.4 area overhead (AOH) is represented by the total number of wrapper boundary cells required for the SoC. Other components of the wrappers such as the controller and the wiring costs are not included because they are similar for both Type 1 and Type 2 wrappers; the boundary cell structures make them unique.

As the cost weight of hardware is increased (increasing β), the total hardware area overhead (columns labeled AOH) decreases while the test application time (columns labeled TAT) increases accordingly. This indicates that as we allow more hardware to be used, more bandwidth-efficient Type 2 wrappers can be used, allowing for a more efficient utilization of bandwidth, hence smaller "rectangles" to pack. Compared to the lower bound defined in Section 6.5.5, the TAT's are on average 13% larger. The area overhead can be reduced considerably without affecting the TAT ($\beta = 0.0$ to 0.5) for all benchmark circuits. This happens when

Table 6.4: TAT for several $B^{i/o}_{max}$. $[\alpha = 1, \beta = 0]$.

$B^{i/o}_{max}$	AOH	TAT	T_{LB}	$\%/T_{LB}$
(Mbps)	p93791noc			
3,200	8,849	923,842	870,079	6.2
6,400	9,303	464,252	435,039	6.7
9,600	9,009	347,378	290,026	19.8
12,800	8,885	235,285	227,978	3.2
$B^{i/o}_{max}$	p22810noc			
3,200	5,584	232,816	203,015	14.7
6,400	5,768	122,091	102,965	18.6
9,600	5,798	102,965	102,965	0.0
12,800	5,798	102,965	102,965	0.0

the Type 1 wrapper is used instead of the Type 2 wrapper for those cores that do not affect the overall TAT.

Table 6.4 shows the resulting AOH and TAT when $B^{i/o}_{max}$ varies from 3.2 *Gbps* to 12.8 *Gbps*, with the objective of minimizing the TAT (i.e. $\alpha = 1, \beta = 0$). This illustrates that without increasing the area overhead, the TAT can be reduced given larger I/O bandwidth, $B^{i/o}_{max}$. This is typically the case because the functional I/O frequency is typically higher than the scan frequency. For the dedicated TAM based approach, TAT reduction can only be achieved by adding TAM wires. $\%/T_{LB}$ in column 5 is calculated as ([column 3] - [column 4]) / [column 4] $\times$ 100%.

Table 6.5 compares our bandwidth sharing approach with the dedicated path (DP) approaches [61, 97, 104, 105]. In the DP approaches, a pair of NoC input and output ports can be used to test only one core at a time. To enable parallel testing, more I/O port pairs are required. Assuming that there is only one I/O port pair, the TAT for DP approach is the sum of each individual core test (sequential testing). Our approach enables parallelism through bandwidth sharing, which proves to be more efficient, with at least 43.1% (when $\alpha = 1, \beta = 0$) smaller TAT. The percentage reduction (%red) is calculated as (DP - SB) / DP $\times$ 100%.

Table 6.5: Test application time of dedicated path (DP) and shared bandwidth (SB) approaches. For SB, $\alpha = 1, \beta = 0$

Channel	Dedicated Path	Shared Bandwidth	%red.
bitwidth, bw	d695noc		
16	49,135	21,768	55.7
32	31,317	17,827	43.1
bitwidth, bw	p93791noc		
16	1,861,439	907,419	51.3
32	1,211,254	464,252	61.7
bitwidth, bw	p22810noc		
16	655,253	229,598	65.0
32	510,954	125,591	75.4

6.7 Conclusion

We have presented a new approach to NoC testing through bandwidth sharing. The test schedule is optimized using a rectangle packing algorithm by optimally assigning to each core a "high gain" bandwidth—the amount of bandwidth that gives a high reduction in TAT. The utilization of two complementary NoC wrappers allow for co-optimization of two most important properties—test application time and area overhead.

It was shown experimentally that it is not always necessary to use the expensive Type 2 wrappers in order to obtain a minimum TAT; the low-cost Type 1 wrappers can be used effectively without compromising the overall TAT. We also evaluated the efficiency of the scheduling algorithm; on average the TAT is less than 13% longer than the theoretical lower bound. Compared to the previously published NoC test scheduling based on dedicated path approach, the proposed bandwidth sharing approach reduces the TAT by an average of 58.7% for the selected case studies.

Chapter 7

Conclusion and Future Work

This chapter first summarizes the work completed in this thesis and then mentions some selected topics that can further enhance the contents of this thesis, as part of the future work.

7.1 Summary of the Thesis

To ensure the performance and correct functionality of a chip, it must be throughly tested for any manufacturing defects and faults caused by statistical manufacturing variations. This thesis presents several methods for the tests of a System-on-Chip, specifically related to the core-based tests. As we consider the test of SoC from the higher perspective, we assume that the design-for-testability (DfT) insertion and the corresponding test generation at the core level can be performed using standard industrial tools. The unique property of the proposed test method is that we try as much as possible to reuse the existing functional components and interconnect for the test purposes. Specifically, we do not add any test access mechanism for the transportation of the test stimuli and responses to the embedded cores.

Chapter 2 explains a test scheduling methodology based on a packet delivery scheduling scheme for core-based testing of System-on-Chips by utilizing the flat functional bus as a test access mechanism. The functional bus is used as a transportation channel for the test stimuli and responses from a tester to the cores under test (CUT). To enable test concurrency, local test buffers are added to all CUT's. In order to limit the buffer area overhead while minimizing the test application time, we propose a packet-based scheduling algorithm called *PAcket Set Scheduling (PASS)*, which finds the complete packet delivery schedule under a given power constraint. The utilization of test packets, consisting of a small number of bits of test data, for test data delivery allow an efficient sharing of bus bandwidth with the help of an effective buffer-based test architecture. The experimental results show that the methodology is highly effective, especially for smaller bus widths, compared to previous approaches that do not use the functional bus.

In Chapter 3, the PASS scheduling scheme for the flat bus is extended for the hierarchical buses of a multiprocessor System-on-Chips (MPSoC). At each core-under-test (CUT), a similar buffer-based test wrapper is used to allow the shared buses to be used for parallel core testing to shorten the test application time, which is the main objective. We then propose a heuristic which uses test configuration graphs to distribute the data transfer load to the system buses in order to maximize efficiency. Parallel core tests are achieved by utilizing packet-based data delivery for the test data transportation to the local buffers, without dedicating communication resources to CUT's. The optimum delivery schedule is achieved by using a packet set-based scheduling methodology. Runs on several modified benchmark circuits provide experimental evidence of the advantages of the proposed methodology. It is applied to both flat bus single processor System-on-Chips (SoC) and hierarchical bus MPSoC's.

Chapter 4 gives a brief description and the characteristics of the SoC which as based on the Network-on-Chip (NoC) interconnect architecture. Based on these characteristics, we then proposed the test wrappers which are suitable for the NoC interconnect architecture, and then followed by a test scheduling algorithm.

Chapter 5 describes the limitation of the IEEE 1500 standard wrapper, which cannot be used for a NoC-based interconnect. This is because of the packet-based transfer mechanism and other functional requirements by the NoC. We describe two NoC-compatible wrappers, which overcome these limitations of the 1500 wrapper. The wrappers (Type 1 and Type 2) complement each other to optimize NoC bandwidth utilization while minimizing the area overhead. The Type 2 wrapper uses larger area overhead to increase bandwidth efficiency, while Type 1 takes advantage of some special configurations which may not require a complex and high-cost wrapper. Two wrapper optimization algorithms are applied to both wrapper designs under channel-bandwidth and test-time constraints, resulting in very little or no increase in the test application time compared to conventional dedicated TAM approaches.

Chapter 6 begins by explaining the limitations of the dedicated path approach, which are used by many previous research on NoC testing. In this approach, a physical path through the NoC routers and interconnects are allocated for the transportation of test data from an external tester to a single core during the whole duration of the core test. This approach unnecessarily limits test concurrency of the embedded cores because a physical channel bandwidth is typically larger than the scan rate of any core-under-test. In this chapter, we propose a bandwidth sharing approach that divides the physical channel bandwidth into multiple smaller virtual channel bandwidths. The test scheduling is performed under the objective of co-optimizing the wrapper area cost and the resulting test application time using two complementary NoC wrappers. Experimental results showed that the area overhead can be optimized (to an extent) without compromising the test application time. Compared to other NoC scheduling approaches based on dedicated paths, our bandwidth sharing approach can reduce the test application time by up to 75.4%.

In this thesis, we have covered the test wrapper designs and test scheduling for a wide range of SoC architectures. By reusing the functional interconnects, our method is already at an advantage compared to others that rely on the use of dedicated TAM's. Furthermore, the proposed test scheduling schemes are also effective in minimizing the test costs—test application time and area cost.

7.2 Future Work

This thesis presented systematic approaches to the test scheduling of various types of SoC's. The first future step should be to implement the proposed test wrappers (both buffer-based wrappers for bus-based SoC's and the Type 1 and Type 2 wrappers for the NoC) at RTL or layout level. The wrapper implementation would serve two purposes. First, to evaluate the actual area costs of the wrappers.

Second, to be used in the case study with an actual SoC test system, which include the test program generation described below.

The PAcket Scheduling Scheme (PASS) and MultiProcessor PASS (MPPASS) presented in Chapters 2 and 3 are evaluated by simulating the test application using a model. The next step toward fully automating the test program generation is to provide a case study where the actual test program can be developed and tested using an actual SoC test system. The test program should use only a few CPU instructions required to transfer the test data between an external device and the CUT, after the CUT is configured in the test mode.

The test scheduling scheme for the MPSoC assumes that the embedded processors designated as testers can access the required test data either from an internal memory or from an external source through memory controllers. We realize that the embedded memory capacity is much too small to be able to accommodate all the test data. An extension to this work would be to remove this assumption during the test scheduling. Since multiprocessor systems with complex buses are rather common in the new SoC systems, it is important to be able to reuse the embedded processors as test resources.

Being able to reuse the embedded processors as test sources means that the I/O bandwidth constraint assumed in Chapter 6 is no longer the critical constraint. In this condition where the test stimuli are generated on-chip, we can consider the problem of scheduling the large internal NoC bandwidth rather than the limited I/O bandwidth.

Acknowledgments

This dissertation is the result of three years of work during which I have had the company and support of many people. I would like to take this opportunity to express my gratitude to all of them. Without their help and advice, this work would not have been possible.

First and foremost, I would like to thank my supervisor, Professor Hideo Fujiwara, for his guidance and support during my graduate studies, and for the invaluable training I received as his student. When I first stepped foot at Nara Institute of Science and Technology, Professor Hideo Fujiwara ensured that my transition as a new member of the Fujiwara Laboratory is a smooth one. During the initial few months as a PhD student, his guidance was most valuable as I work to establish a foundation for my research. During the years, Professor Hideo Fujiwara always provided a motivational, enthusiastic, and critical atmosphere during the many discussions we had. Without exception, he was always there to relate back to the bigger picture whenever I get lost in the jungle of technical details. It was a great pleasure for me to have had the opportunity to complete my research and dissertation under his supervision.

I would like to especially thank Assistant Professor Tomokazu Yoneda for his close guidance and advice during every discussion. I was fortunate to have had the opportunity to work very closely with Assistant Professor Tomokazu Yoneda. I would like to thank him for his patience in meticulously proofreading and commenting every draft of every conference and journal papers resulting from my research. Not forgetting his contribution in improving the earlier drafts of this dissertation. Thank you for your supervision and friendship.

I am grateful to Professor Alex Orailoglu of the University of California San Diego, U.S.A. for his valuable comments during our research collaboration and on the drafts of the conference and journal papers. He has given extremely insightful advice aimed at making me target higher and higher in doing research.

I would like to extend my gratitude to Professor Krishnendu Chakrabarty of Duke University, U.S.A. for his advice and motivational support of my research. Contributions from Professor Chakrabarty, who is one of the leading researchers in the field of system-on-chip testing, was of great value and influence in steering my research in the right direction.

I would like to thank the members of the thesis committee, Professor Yasuhiko Nakashima and Associate Professor Michiko Inoue, who monitored my work and took great effort in reading and providing me with valuable comments on earlier versions of this dissertation. A special appreciation goes to Associate Professor Michiko Inoue for her constructive criticisms during the presentations and progress report sessions in the laboratory.

I am indebted to Assistant Professor Satoshi Ohtake for his friendly discussions, comments, and continuous efforts in providing the necessary networking and computing infrastructure in the laboratory. These hardware and software resources were absolutely essential in the success of all the experimental works.

I would also like to thank Dr. Ilia Polian of Albert-Ludwigs-University of Freiburg, Germany and Associate Professor Erik Larsson of Linköpings Universitet, Sweden for their comments and contributions.

I would also like to extend my gratitude to Ms. Saeko Ono and Ms. Aiko Sato for helping me with various tasks and official correspondence in the laboratory.

The members of Fujiwara Laboratory gave me the feeling of being at home at work. Their support and companion makes the everyday work at the laboratory more pleasant and enjoyable. Dr. Kazuko Kambe, Dr. Virendra Singh, Dr. Zhiqiang You, Dr. Hiroyuki Iwata, Mr. Vorayos Thongtan, Mr. Thomas Edison Yu, Ms. Elena Davydova Dikkanen, Mr. Chi-Che Hsieh, thank you for being my friends and colleagues. Special thanks to Dr. Chia Yee Ooi and Dr. Masato Nakazato for their help with settling my life down when I first arrived in Japan. Special thanks also to Dr. Thomas Clouqueur and Dr. Yuuki Yoshikawa for being my tennis partners, best friends and colleagues. Thanks also to all other past and present members of Fujiwara Laboratory for making my stay, within the confines of the laboratory, pleasurable. Thank you for your friendship.

This work was supported in part by 21st Century Center of Excellence (COE) Program (Ubiquitous Networked Media Computing) and in part by Japan Society for the Promotion of Science (JSPS) under Grants-in-Aid for Scientific Research and Young Scientists under the grant for activity, education and research. I express my gratitude to all these sponsoring organizations.

I am grateful to the Ministry of Education, Culture, Sports, Science and Technology (MEXT), Government of Japan for supporting me with a Japanese government scholarship throughout my course of study. I am also grateful to the Universiti Teknologi Petronas for providing me a chance to pursue doctoral degree at NAIST, Japan.

I am also grateful to my family members for their love and moral support during these challenging three years. I am especially grateful to my beautiful wife who has made great many sacrifices for the success of my PhD. She was always there to cheer me up when I needed a boost of moral support. Your support is most valued and appreciated.

And to my son for being a motivation.

I profusely thank you all.

Bibliography

[1] Erik Jan Marinissen, Vikram Iyengar, and Krishnendu Chakrabarty. A set of benchmarks fo modular testing of SOCs. *ITC'02: Proc. International Test Conference*, 00:519–528, 2002.

[2] International technology roadmap for semiconductors (ITRS 2007). http://www.itrs.net/Links/2007ITRS/Home2007.htm, Access date: June 2008.

[3] Erik Jan Marinissen, Rohit Kapur, Maurice Lousberg, Teresa McLaurin, Mike Ricchetti, and Yervant Zorian. On IEEE P1500's standard for embedded core test. *J. Electron. Test.*, 18(4-5):365–383, 2002.

[4] 1500 IEEE standard testability method for embedded core-based integrated circuits. http://grouper.ieee.org/groups/1500/index.html.

[5] Erik Larsson. *Chapter 8, Introduction to Advanced System-on-Chip Test Design and Optimization.* Frontiers in Electronic Testing, Vol. 29. Springer Publishing, 2005.

[6] Anders Larsson, Erik Larsson, Petru Eles, and Zebo Peng. Optimization of a bus-based test data transportation mechanism in system-on-chip. In *DSD '05: Proceedings of the 8th Euromicro Conference on Digital System Design*, pages 403–411, Washington, DC, USA, 2005. IEEE Computer Society.

[7] Alexandre M. Amory, Kees Goossens, Erik Jan Marinissen, Marcelo Lubaszewski, and Fernando Moraes. Wrapper design for the reuse of networks-on-chip as test access mechanism. In *ETS '06: Proceedings of the Eleventh IEEE European Test Symposium*, pages 213–218, Washington, DC, USA, 2006. IEEE Computer Society.

[8] Yervant Zorian, Erik Jan Marinissen, and Sujit Dey. Testing embedded-core based system chips. In *ITC '98: Proceedings of the 1998 IEEE International*

Test Conference, page 130, Washington, DC, USA, 1998. IEEE Computer Society.

[9] Erik Jan Marinissen, Robert G. J. Arendsen, Gerard Bos, Hans Dingemanse, Maurice Lousberg, and Clemens Wouters. A structured and scalable mechanism for test access to embedded reusable cores. In *ITC '98: Proceedings of the 1998 IEEE International Test Conference*, pages 284–293, Washington, DC, USA, 1998. IEEE Computer Society.

[10] Anuja Sehgal, Vikram Iyengar, Mark D. Krasniewski, and Krishnendu Chakrabarty. Test cost reduction for SOCs using virtual TAMs and lagrange multipliers. In *DAC '03: Proceedings of the 40th conference on Design automation*, pages 738–743, New York, NY, USA, 2003. ACM.

[11] Tomokazu Yoneda and Hideo Fujiwara. A DFT method for core-based systems-on-a-chip based on consecutive testability. In *ATS '01: Proceedings of the 10th Asian Test Symposium*, pages 193–198, Washington, DC, USA, 2001. IEEE Computer Society.

[12] Vikram Iyengar, Krishnendu Chakrabarty, and Erik Jan Marinissen. Test wrapper and test access mechanism co-optimization for system-on-chip. *J. Electron. Test.*, 18(2):213–230, 2002.

[13] Yu Huang, Sudhakar M. Reddy, Wu-Tung Cheng, Paul Reuter, Nilanjan Mukherjee, Chien-Chung Tsai, Omer Samman, and Yahya Zaidan. Optimal core wrapper width selection and SoC test scheduling based on 3-D bin packing algorithm. In *ITC '02: Proceedings of the 2002 IEEE International Test Conference*, pages 74–83, Washington, DC, USA, 2002. IEEE Computer Society.

[14] Erik Larsson and Hideo Fujiwara. System-on-chip test scheduling with reconfigurable core wrappers. *IEEE Trans. Very Large Scale Integr. Syst.*, 14(3):305–309, 2006.

[15] Tomokazu Yoneda, Kimihiko Masuda, and Hideo Fujiwara. Power-constrained test scheduling for multi-clock domain SoCs. In *DATE '06: Proceedings of the conference on Design, automation and test in Europe*, pages 297–302, 3001 Leuven, Belgium, Belgium, 2006. European Design and Automation Association.

[16] Anuja Sehgal, Vikram Iyengar, and Krish Chakrabarty. SOC test planning

using virtual test access architectures. *IEEE Trans. Very Large Scale Integr. Syst.*, 12(12):1263–1276, 2004.

[17] Yu Xia, Malgorzata Chrzanowska-Jeske, Benyi Wang, and Marcin Jeske. Using a distributed rectangle bin-packing approach for core-based SoC test scheduling with power constraints. In *ICCAD '03: Proceedings of the 2003 IEEE/ACM international conference on Computer-aided design*, pages 100–105, Washington, DC, USA, 2003. IEEE Computer Society.

[18] Chih-Pin Su and Cheng-Wen Wu. A graph-based approach to power-constrained SoC test scheduling. *J. Electron. Test.*, 20(1):45–60, 2004.

[19] Julien Pouget, Erik Larsson, and Zebo Peng. Multiple-constraint driven system-on-chip test time optimization. *J. Electron. Test.*, 21(6):599–611, 2005.

[20] Krish Chakrabarty, Vikram Iyengar, and M. D. Krasniewski. Test planning for modular testing of hierarchical SOCs. *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, 24(3):435–448, 2005.

[21] Sandeep Kumar Goel and Erik Jan Marinissen. SOC test architecture design for efficient utilization of test bandwidth. *ACM Trans. Des. Autom. Electron. Syst.*, 8(4):399–429, 2003.

[22] Tomokazu Yoneda and Hideo Fujiwara. Design for consecutive testability of system-on-a-chip with built-in self testable cores. *J. Electron. Test.*, 18(4-5):487–501, 2002.

[23] Peter Harrod. Testing reusable IP - a case study. In *ITC '99: Proceedings of the 1999 IEEE International Test Conference*, pages 493–498, Washington, DC, USA, 1999. IEEE Computer Society.

[24] C. A. Papachristou, F. Martin, and M. Nourani. Microprocessor based testing for core-based system on chip. In *DAC '99: Proceedings of the 36th ACM/IEEE conference on Design automation*, pages 586–591, New York, NY, USA, 1999. ACM.

[25] Angela Krstic, Wei-Cheng Lai, Kwang-Ting Cheng, Li Chen, and Sujit Dey. Embedded software-based self-test for programmable core-based designs. *IEEE Des. Test*, 19(4):18–27, 2002.

[26] Jing-Reng Huang, Madhu K. Iyer, and Kwang-Ting Cheng. A self-test methodology for IP cores in bus-based programmable SoCs. In *VTS '01:*

Proceedings of the 19th IEEE VLSI Test Symposium, pages 198–203, Washington, DC, USA, 2001. IEEE Computer Society.

[27] Anders Larsson, Erik Larsson, Petru Eles, and Zebo Peng. SOC test scheduling with test set sharing and broadcasting. In *ATS '05: Proceedings of the 14th Asian Test Symposium*, pages 162–169, Washington, DC, USA, 2005. IEEE Computer Society.

[28] Sungbae Hwang and J. A. Abraham. Microprocessor based testing for core-based system on chip. In *Proceedings of the 14th IEEE International ASIC/SOC Conference*, pages 215–219, New York, NY, USA, 2001. IEEE.

[29] Sandeep Koranne and Vikram Iyengar. On the use of k-tuples for SoC test schedule representation. In *ITC '02: Proceedings of the 2002 IEEE International Test Conference*, pages 539–548, Washington, DC, USA, 2002. IEEE Computer Society.

[30] Krishnendu Chakrabarty. Design of system-on-a-chip test access architectures using integer linear programming. In *VTS '00: Proceedings of the 18th IEEE VLSI Test Symposium (VTS'00)*, page 127, Washington, DC, USA, 2000. IEEE Computer Society.

[31] Krishnendu Chakrabarty. Test scheduling for core-based systems using mixed-integer linear programming. *IEEE Transactions on Computer-Aided Design*, 19(10):1163–1174, 2000.

[32] Vikram Iyengar and Krishnendu Chakrabarty. Test bus sizing for system-on-a-chip. *IEEE Trans. Comput.*, 51(5):449–459, 2002.

[33] Wei Zou, Sudhakar M. Reddy, Irith Pomeranz, and Yu Huang. SOC test scheduling using simulated annealing. In *VTS '03: Proceedings of the 21st IEEE VLSI Test Symposium*, pages 325–330, Washington, DC, USA, 2003. IEEE Computer Society.

[34] Vikram Iyengar and Krish Chakrabarty. Software-based self-testing methodology for processor cores. *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, 21(9):1088–1094, 2002.

[35] Erik Larsson, K. Arvidsson, Hideo Fujiwara, and Zebo Peng. Efficient test solutions for core-based designs. *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, 23(5):758–775, 2004.

[36] Masahide Miyazaki, Toshinori Hosokawa, Hiroshi Date, Michiaki Muraoka,

and Hideo Fujiwara. A DFT selection method for reducing test application time of system-on-chips. In *ATS '00: Proceedings of Asian Test Symposium*, volume 0, pages 412–417, Los Alamitos, CA, USA, 2003. IEEE Computer Society.

[37] Qiang Xu and Nicola Nicolici. Multi-frequency test access mechanism design for modular SoC testing. In *ATS '04: Proceedings of the 13th Asian Test Symposium*, pages 2–7, Washington, DC, USA, 2004. IEEE Computer Society.

[38] Dan Zhao and Shambhu Upadhyaya. Power constrained test scheduling with dynamically varied TAM. In *VTS '03: Proceedings of the 21st IEEE VLSI Test Symposium*, pages 273–280, Washington, DC, USA, 2003. IEEE Computer Society.

[39] Anuja Sehgal and Krishnendu Chakrabarty. Efficient modular testing of SOCs using dual-speed TAM architectures. In *DATE '04: Proceedings of the conference on Design, automation and test in Europe*, pages 422–427, Washington, DC, USA, 2004. IEEE Computer Society.

[40] Erik Larsson, Julien Pouget, and Zebo Peng. Defect-aware SoC test scheduling. In *VTS '04: Proceedings of the 22nd IEEE VLSI Test Symposium*, pages 361–366, Washington, DC, USA, 2004. IEEE Computer Society.

[41] Zhiyuan He, Zebo Peng, and Petru Eles. Power constrained and defect-probability driven SoC test scheduling with test set partitioning. In *DATE '06: Proceedings of the conference on Design, automation and test in Europe*, pages 291–296, 3001 Leuven, Belgium, Belgium, 2006. European Design and Automation Association.

[42] Richard M. Chou, Kewal K. Saluja, and Vishwani D. Agrawal. Scheduling tests for VLSI systems under power constraints. *IEEE Trans. Very Large Scale Integr. Syst.*, 5(2):175–185, 1997.

[43] Krishnendu Chakrabarty. Design of system-on-a-chip test access architectures under place-and-route and power constraints. In *DAC '00: Proceedings of the 37th conference on Design automation*, pages 432–437, New York, NY, USA, 2000. ACM.

[44] Vikram Iyengar and Krishnendu Chakrabarty. Precedence-based, preemptive, and power-constrained test scheduling for system-on-a-chip. In *VTS*

'01: Proceedings of the 19th IEEE VLSI Test Symposium, pages 368–373, Washington, DC, USA, 2001. IEEE Computer Society.

[45] E. Larsson and Z. Peng. An integrated system-on-chip test framework. In *DATE '01: Proceedings of the conference on Design, automation and test in Europe*, pages 138–144, Piscataway, NJ, USA, 2001. IEEE Press.

[46] Mehrdad Nourani and James Chin. Power-time tradeoff in test scheduling for SoCs. In *ICCD '03: Proceedings of the 21st International Conference on Computer Design*, pages 548–553, Washington, DC, USA, 2003. IEEE Computer Society.

[47] Sandeep Kumar Goel and Erik Jan Marinissen. Layout-driven SoC test architecture design for test time and wire length minimization. In *DATE '03: Proceedings of the conference on Design, Automation and Test in Europe*, pages 738–743, Washington, DC, USA, 2003. IEEE Computer Society.

[48] Anuja Sehgal, Sandeep Kumar Goel, Erik Jan Marinissen, and Krishnendu Chakrabarty. Hierarchy-aware and area-efficient test infrastructure design for core-based system chips. In *DATE '06: Proceedings of the conference on Design, automation and test in Europe*, pages 285–290, 3001 Leuven, Belgium, Belgium, 2006. European Design and Automation Association.

[49] J. Li, H. Huang, J. Chen, C. Su, C. Wu, C. Cheng, S. Chen, C. Hwang, and H. Lin. A hierarchical test scheme for system-on-chip designs. In *DATE '02: Proceedings of the conference on Design, automation and test in Europe*, pages 486–490, Washington, DC, USA, 2002. IEEE Computer Society.

[50] Vikram Iyengar, Krishnendu Chakrabarty, Mark D. Krasniewski, and Gopind N. Kumar. Design and optimization of multi-level TAM architectures for hierarchical SOCs. In *VTS '03: Proceedings of the 21st IEEE VLSI Test Symposium*, pages 299–312, Washington, DC, USA, 2003. IEEE Computer Society.

[51] Yu Huang, Wu-Tung Cheng, Chien-Chung Tsai, Nilanjan Mukherjee, Omer Samman, Yahya Zaidan, and Sudhakar M. Reddy. Resource allocation and test scheduling for concurrent test of core-based SoC design. In *ATS '01: Proceedings of the 10th Anniversary Compendium of Papers from Asian Test Symposium 1992-2001*, pages 361–366, Washington, DC, USA, 2001.

IEEE Computer Society.

[52] Tsuyoshi Shinogi, Yuki Yamada, Terumine Hayashi, Tomohiro Yoshikawa, and Shinji Tsuruoka. Between-core vector overlapping for test cost reduction in core testing. *ATS '03: Proceedings of Asian Test Symposium*, 0:268–273, 2003.

[53] Erika Cota, Luigi Carro, Marcelo Lubaszewski, and Alex Orailoglu. Test planning and design space exploration in a core-based environment. In *DATE '02: Proceedings of the conference on Design, automation and test in Europe*, pages 478–443, Washington, DC, USA, 2002. IEEE Computer Society.

[54] Fawnizu Azmadi Hussin, Tomokazu Yoneda, Alex Orailoglu, and Hideo Fujiwara. Power-constrained SoC test schedules through utilization of functional buses. In *24th IEEE International Conference on Computer Design (ICCD'06)*, 2006.

[55] Fawnizu Azmadi Hussin, Tomokazu Yoneda, Alex Orailoglu, and Hideo Fujiwara. Scheduling power-constrained tests through the SoC functional bus. *IEICE Transactions on Information and Systems*, E91-D(3):736–746, Mar. 2008.

[56] Li Chen and Sujit Dey. Software-based self-testing methodology for processor cores. *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, 20(3):369–380, 2001.

[57] Madhu K. Iyer and Kwang-Ting Cheng. Software-based weighted random testing for IP cores in bus-based programmable SoCs. In *Proceedings of the VLSI Test Symposium*, pages 139–144, 2002.

[58] Wei-Cheng Lai and Kwang-Ting Cheng. Instruction-level DFT for testing processor and IP cores in system-on-a-chip. In *DAC '01: Proceedings of the 38th conference on Design automation*, pages 59–64, New York, NY, USA, 2001. ACM.

[59] Chouki Aktouf. A complete strategy for testing an on-chip multiprocessor architecture. *IEEE Des. Test*, 19(1):18–28, 2002.

[60] B. Vermeulen, J. Dielissen, K. Goossens, and K. Ciordas. Bringing communication networks on chip: Test and verification implications. *IEEE Communications Magazine*, 41(9):74–81, 2003.

[61] E. Cota, M. Kreutz, C. A. Zeferino, L. Carro, M. Lubaszewski, and A. Susin. The impact of NoC reuse on the testing of core-based systems. In *VTS '03: Proceedings of the 21st IEEE VLSI Test Symposium*, page 128, Washington, DC, USA, 2003. IEEE Computer Society.

[62] Alexandre M. Amory, Érika Cota, Marcelo Lubaszewski, and Fernando G. Moraes. Reducing test time with processor reuse in network-on-chip based systems. In *SBCCI '04: Proceedings of the 17th symposium on Integrated circuits and system design*, pages 111–116, New York, NY, USA, 2004. ACM.

[63] Chunsheng Liu, Zach Link, and D. K. Pradhan. Reuse-based test access and integrated test scheduling for network-on-chip. In *DATE '06: Proceedings of the conference on Design, automation and test in Europe*, pages 303–308, 3001 Leuven, Belgium, Belgium, 2006. European Design and Automation Association.

[64] M. Nahvi and A. Ivanov. Indirect test architecture for SoC testing. *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, 23(7):1128–1142, 2004.

[65] David Flynn. AMBA: Enabling reusable on-chip designs. *IEEE Micro*, 17(4):20–27, 1997.

[66] Joep Aerts and Erik Jan Marinissen. Scan chain design for test time reduction in core-based ICs. In *ITC '98: Proceedings of the 1998 IEEE International Test Conference*, pages 448–457, Washington, DC, USA, 1998. IEEE Computer Society.

[67] Nicola Nicolici and Bashir M. Al-Hashimi. *Chapter 2, Power-Constrained Testing of VLSI Circuits.* Kluwer Academic Publishers, 2003.

[68] Sandeep Kumar Goel and Erik Jan Marinissen. Effective and efficient test architecture design for SOCs. In *ITC '02: Proceedings of the 2002 IEEE International Test Conference*, page 529, Washington, DC, USA, 2002. IEEE Computer Society.

[69] Yervant Zorian. Test requirements for embedded core-based systems and IEEE P1500. In *Proceedings of the IEEE International Test Conference*, pages 191–199, Washington, DC, USA, 1997. IEEE Computer Society.

[70] Anuja Sehgal, Sandeep Kumar Goel, Erik Jan Marinissen, and Krishnendu

Chakrabarty. IEEE P1500-compliant test wrapper design for hierarchical cores. In *ITC '04: Proceedings of the International Test Conference on International Test Conference*, pages 1203–1212, Washington, DC, USA, 2004. IEEE Computer Society.

[71] Qiang Xu and Nicola Nicolici. Wrapper design for testing IP cores with multiple clock domains. In *DATE '04: Proceedings of the conference on Design, automation and test in Europe*, pages 416–421, Washington, DC, USA, 2004. IEEE Computer Society.

[72] Sandeep Kumar Goel and Erik Jan Marinissen. Cluster-based test architecture design for system-on-chip. In *VTS '02: Proceedings of the 20th IEEE VLSI Test Symposium*, pages 259–264, Washington, DC, USA, 2002. IEEE Computer Society.

[73] Luca Benini and Giovanni De Micheli. Networks on chips: A new SoC paradigm. *Computer*, 35(1):70–78, 2002.

[74] Fawnizu Azmadi Hussin, Tomokazu Yoneda, Alex Orailoglu, and Hideo Fujiwara. Core-based testing of multiprocessor system-on-chips utilizing hierarchical functional buses. In *ASP-DAC '07: Proceedings of the 2007 conference on Asia South Pacific design automation*, pages 720–725, Washington, DC, USA, 2007. IEEE Computer Society.

[75] Pierre Guerrier and Alain Greiner. A generic architecture for on-chip packet-switched interconnections. In *DATE '00: Proceedings of the conference on Design, automation and test in Europe*, pages 250–256, New York, NY, USA, 2000. ACM.

[76] Faraydon Karim, Anh Nguyen, Sujit Dey, and Ramesh Rao. On-chip communication architecture for OC-768 network processors. In *DAC '01: Proceedings of the 38th conference on Design automation*, pages 678–683, New York, NY, USA, 2001. ACM.

[77] I. Saastamoinen, D. Sigüenza-Tortosa, and J. Nurmi. Interconnect IP node for future system-on-chip designs. In *DELTA '02: Proceedings of the The First IEEE International Workshop on Electronic Design, Test and Applications (DELTA '02)*, pages 116–122, Washington, DC, USA, 2002. IEEE Computer Society.

[78] Sashi Kumar, Axel Jantsch, Juha-Pekka Soininen, Martti Forsell, Mikael

Millberg, Johny Oberg, Kari Tiensyrja, , and Ahmed Hemani. A network on chip architecture and design methodology. In *ISVLSI '02: Proceedings of the IEEE Computer Society Annual Symposium on VLSI*, pages 105–112, Washington, DC, USA, 2002. IEEE Computer Society.

[79] E. Rijpkema, K. G. W. Goossens, A. Radulescu, J. Dielissen, J. van Meerbergen, P. Wielage, and E. Waterlander. Trade offs in the design of a router with both guaranteed and best-effort services for networks on chip. In *DATE '03: Proceedings of the conference on Design, Automation and Test in Europe*, pages 350—355, Washington, DC, USA, 2003. IEEE Computer Society.

[80] Andrei Radulescu, John Dielissen, Santiago Gonzalez Pestana, Om Prakash Gangwal, Edwin Rijpkema, Paul Wielage, , and Kees Goossens. An efficient on-chip NI offering guaranteed services, shared-memory abstraction, and flexible network configuration. *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, 24(1):4–17, 2005.

[81] Cesar Albenes Zeferino and Altamiro Amadeu Susin. SoCIN: A parametric and scalable network-on-chip. In *SBCCI '03: Proceedings of the 16th symposium on Integrated circuits and systems design*, pages 169–174, Washington, DC, USA, 2003. IEEE Computer Society.

[82] Daniel Wiklund and Dake Liu. SoCBUS: Switched network on chip for hard real time embedded systems. In *IPDPS '03: Proceedings of the 17th International Symposium on Parallel and Distributed Processing*, page 78.1, Washington, DC, USA, 2003. IEEE Computer Society.

[83] Matteo Dall'Osso, Gianluca Biccari, Luca Giovannini, Davide Bertozzi, and Luca Benini. xpipes: a latency insensitive parameterized network-on-chip architecture for multi-processor SoCs. In *ICCD '03: Proceedings of the 21st International Conference on Computer Design*, pages 536–539, Washington, DC, USA, 2003. IEEE Computer Society.

[84] Mikael Millberg, Erland Nilsson, Rikard Thid, and Axel Jantsch. Guaranteed bandwidth using looped containers in temporally disjoint networks within the nostrum network on chip. In *DATE '04: Proceedings of the conference on Design, automation and test in Europe*, pages 890–895, Washington, DC, USA, 2004. IEEE Computer Society.

[85] Evgeny Bolotin, Israel Cidon, Ran Ginosar, and Avinoam Kolodny. QNoC: QoS architecture and design process for network on chip. *J. Syst. Archit.: The EUROMICRO Journal*, 50(2-3):105–128, 2004.

[86] Fernando Moraes, Ney Calazans, Aline Mello, Leandro Möller, and Luciano Ost. HERMES: an infrastructure for low area overhead packet-switching networks on chip. *Integr. VLSI J.*, 38(1):69–93, 2004.

[87] John Bainbridge and Steve Furber. Chain: A delay-insensitive chip area interconnect. *IEEE Micro*, 22(5):16–23, 2002.

[88] Andrew Lines. Asynchronous interconnect for synchronous SoC design. *IEEE Micro*, 24(1):32–41, 2004.

[89] Edith Beigne, Fabien Clermidy, Pascal Vivet, Alain Clouard, and Marc Renaudin. An asynchronous NoC architecture providing low latency service and its multi-level design framework. In *ASYNC '05: Proceedings of the 11th IEEE International Symposium on Asynchronous Circuits and Systems*, pages 54–63, Washington, DC, USA, 2005. IEEE Computer Society.

[90] Tobias Bjerregaard and Jens Sparso. A router architecture for connection-oriented service guarantees in the MANGO clockless network-on-chip. In *DATE '05: Proceedings of the conference on Design, Automation and Test in Europe*, pages 1226–1231, Washington, DC, USA, 2005. IEEE Computer Society.

[91] AMBA AXI protocol specification, 2004.

[92] Open core protocol specification, release 2.1a, 2005.

[93] Erik Jan Marinissen, Sandeep Kumar Goel, and Maurice Lousberg. Wrapper design for embedded core test. In *ITC '00: Proceedings of the 2000 IEEE International Test Conference*, pages 911–920, Washington, DC, USA, 2000. IEEE Computer Society.

[94] Fawnizu Azmadi Hussin, Tomokazu Yoneda, and Hideo Fujiwara. Optimization of NoC wrapper design under bandwidth and test time constraints. In *ETS '07: Proceedings of the 12th IEEE European Test Symposium*, pages 35–42, Washington, DC, USA, 2007. IEEE Computer Society.

[95] Fawnizu Azmadi Hussin, Tomokazu Yoneda, and Hideo Fujiwara. NoC-compatible wrapper design and optimization under channel bandwidth and test time constraints. *IEICE Transactions on Information and Systems*,

E91-D(7):2008–2017, July 2008.

[96] Alexandre M. Amory, Kees Goossens, Erik J. Marinissen, M. Lubaszewski, and F. Moraes. Wrapper design for the reuse of a bus, network-on-chip, or other functional interconnect as test access mechanism. *IET Computers & Digital Techniques*, 1(3):197–206, 2007.

[97] Érika Cota, Luigi Carro, and Marcelo Lubaszewski. Reusing an on-chip network for the test of core-based systems. *ACM Trans. Des. Autom. Electron. Syst.*, 9(4):471–499, 2004.

[98] Alexandre M. Amory Eduardo Briao Erika Cota Marcelo Lubaszewski and Fernando G. Moraes. Wrapper design for embedded core test. In *ITC '05: Proceedings of the 2005 IEEE International Test Conference*, pages 591–599, Washington, DC, USA, 2005. IEEE Computer Society.

[99] Cristian Grecu, Partha Pande, Andre Ivanov, and Res Saleh. BIST for network-on-chip interconnect infrastructures. In *VTS '06: Proceedings of the 24th IEEE VLSI Test Symposium*, pages 30–35, Washington, DC, USA, 2006. IEEE Computer Society.

[100] Xuan-Tu Tran, Jean Durupt, Francois BERTRAND Bertrand, Vincent Beroulle, and Chantal Robach. A DFT architecture for asynchronous networks-on-chip. In *ETS '06: Proceedings of the Eleventh IEEE European Test Symposium*, pages 219–224, Washington, DC, USA, 2006. IEEE Computer Society.

[101] A. Khoche. Test resource partitioning for scan architectures using bandwidth matching. In *Digest of Workshop on Test Resource Partitioning*, pages 1.4.1–1.4.8, Washington, DC, USA, 2002. IEEE Computer Society.

[102] Chunsheng Liu and Vikram Iyengar. Test scheduling with thermal optimization for network-on-chip systems using variable-rate on-chip clocking. In *DATE '06: Proceedings of the conference on Design, automation and test in Europe*, pages 652–657, 3001 Leuven, Belgium, Belgium, 2006. European Design and Automation Association.

[103] Chunsheng Liu, Vikram Iyengar, and D. K. Pradhan. Thermal-aware testing of network-on-chip using multiple-frequency clocking. In *VTS '06: Proceedings of the 24th IEEE VLSI Test Symposium*, pages 46–51, Washington, DC, USA, 2006. IEEE Computer Society.

[104] Chunsheng Liu, Hamid Sharif, Erika Cota, and D. K. Pradhan. Test scheduling for network-on-chip with bist and precedence constraints. In *ITC '04: Proceedings of the International Test Conference*, pages 1369–1378, Washington, DC, USA, 2004. IEEE Computer Society.

[105] Chunsheng Liu, Zach Link, and D. K. Pradhan. Reuse-based test access and integrated test scheduling for network-on-chip. In *DATE '06: Proceedings of the conference on Design, automation and test in Europe*, pages 303–308, 3001 Leuven, Belgium, Belgium, 2006. European Design and Automation Association.

[106] Vikram Iyengar, Krishnendu Chakrabarty, and Erik Jan Marinissen. On using rectangle packing for SoC wrapper/TAM co-optimization. In *VTS '02: Proceedings of the 20th IEEE VLSI Test Symposium*, pages 253–258, Washington, DC, USA, 2002. IEEE Computer Society.

[107] Fawnizu Azmadi Hussin, Tomokazu Yoneda, and Hideo Fujiwara. Area overhead and test time co-optimization through NoC bandwidth sharing. In *ATS '07: Proceedings of the 16th Asian Test Symposium*, pages 459–462, Washington, DC, USA, 2007. IEEE Computer Society.

[108] Fawnizu Azmadi Hussin, Tomokazu Yoneda, and Hideo Fujiwara. On NoC bandwidth sharing for the optimization of area cost and test application time. *IEICE Transactions on Information and Systems*, E91-D(7):1999–2007, July 2008.