UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

ALEXANDRE DE MORAIS AMORY

# Lógica e Escalonamento de Teste para Sistemas com Redes Intra-Chip Baseadas em Topologia de Malha

Tese apresentada como requisito parcial
para a obtenção do grau de
Doutor em Ciência da Computação

Prof. Dr. Marcelo Lubaszewski
Orientador

Prof. Dr. Fernando Gehm Moraes
Co-orientador

Porto Alegre, Agosto 2007

**CIP – CATALOGAÇÃO NA PUBLICAÇÃO**

Amory, Alexandre de Morais

Lógica e Escalonamento de Teste para Sistemas com Redes Intra-Chip Baseadas em Topologia de Malha / Alexandre de Morais Amory. – Porto Alegre: Programa de Pós-Graduação em Computação, 2007.

178 f.: il.

Tese (doutorado)– Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2007. Orientador: Marcelo Lubaszewski; Co-orientador: Fernando Gehm Moraes.

1. Teste de sistemas intra-chip. 2. Lógica envoltória de teste. 3. Escalonamento de teste. 4. Redes intra-chip. I. Lubaszewski, Marcelo. II. Moraes, Fernando Gehm. III. Título.

*à vó Carminha*

# AGRADECIMENTOS

O período do doutorado é longo e cansativo. Percorrer esse caminho com o apoio que tive certamente tornou a caminhada mais simples. Inicialmente gostaria de agradecer o apoio financeiro da Capes e CNPq; não agradeço somente a ajuda finaceira, mas também a oportunidade de fazer estágio no exterior, que realmente expandiu meus horizontes. Falando em novos horizontes no exterior, gostaria de agradecer a Erik Jan Marinissen e Kees Goossens da Philips Research Labs pela oportunidade e aprendizado.

Nao é por falta de imaginação, eu sei que já usei essa passagem de Gandhi antes, mas ainda não consegui achar uma passagem que representasse melhor o meu agradecimento aos meus orientadores Marcelo Lubaszewski e Fernando Moraes: *"... I have always felt that the true text-book for the pupil is his teacher. I remember very little that my teachers taught me from books, but I have even now a clear recollection of the things they taught me independently of books"*. O conhecimento 'dos livros' é obviamente necessário para a conclusão do doutorado. Isso definitivamente recebi nas nossas inumeras discuções, brain-storms, e revisões de texto. Entretanto, o conhecimento técnico, principalmente nessa área ligado à tecnologia de ponta, fica defasado rapidamente. Porém, o ensinamento mencionado na frase de Gandi não deteriora. Pode passar muito tempo mas, mesmo que involuntariamente, ainda estarei usando esse conhecimento, e eventualmente passando adiante.

Doutarado muitas vezes envolve muito stress; isso é inevitável. Quem melhor para aliviar (ou melhor, evitar) esse stress do que com seus amigos, tomando uma cervejinha ? Felizmente tive bons amigos para recorrer como: o Fabiano, amigo de longa data; o pessoal do GAPH: Ost [1], Möller, Ewerson, Edson, e Castanha; o pessoal do GME: Gustavo, Lazzari, Edgard, Zé, Júlio, Paulo, Renato e Brião; People from Eindhoven: Tobias, Sylvain, Ram, Patrick, Vlado, Goran, Rodrigo, Samir, and Saurin; e mais recentemente tem o pessoal do CEITEC. Obrigado! vocês tornaram minha caminhada mais tranquila.

Muito bem, amigos ajudam e muito a aliviar/evitar stress. Mas e quando isso não é o suficiente ? A quem recorrer nos momentos bons e também nos "não tão bons assim"? recorrer tanto na hora certa quanto na hora errada ? No meu caso eu recorria as meninas da minha vida: Ariela, Duda, Camila, Leticia e Sonia. Tenho muito a agradeçer a elas. Não sei nem por onde começar. Sei que também tenho muito que me desculpar, principalente pela ausência pois essa coisinha chamada "Doutorado" consume muito. À Leticia, que esteve comigo esse tempo todo, obrigado pela paciência em aguentar meu mau humor! Obrigado pela ajuda! Obrigado pelo apoio! Obrigado pelo carinho! ... Obrigado por tudo! Enfim, terminou!

[1] Viu só! não me esqueci de ti dessa vez !!! Não pude perder a oportunidade :-P

# TABLE OF CONTENTS

# LIST OF ABBREVIATIONS AND ACRONYMS

| | |
|---|---|
| ASIC | Application-Specific Integrated Circuit |
| ATE | Automatic Test Equipment |
| ATPG | Automatic Test Pattern Generation |
| AXI | Advanced eXtensible Interface |
| BE | Best-Effort |
| BIST | Built-in Self-Test |
| CUT | Core-Under-Test |
| DfT | Design-for-Test |
| DSM | Deep Sub-Micron |
| DTL | Device Transaction Level |
| FIFO | First-In First-Out |
| GT | Guaranteed Throughput |
| IP | Intellectual Property |
| LFSR | Linear feedback Shift Register |
| MISR | Multiple Input Signature Register |
| NI | Network Interface |
| NoC | Network-on-Chip |
| OCP | Open Core Protocol |
| QoS | Quality of Service |
| SoC | System-on-Chip |
| TAM | Test Access Mechanism |
| TRP | Test Resource Partition |
| UDL | User-Defined Logic |
| vTAM | virtual Test Access Mechanism |

# LIST OF SYMBOLS

| | |
|---|---|
| $i_g^p$ | the input test port of the module $g$. Page 77. |
| $o_g^p$ | the output test port of the module $g$. Page 77. |
| $p_g$ | the number of test patterns of the module $g$. Page 77. |
| $i_g$ | the number of functional input terminals of the module $g$. Page 77. |
| $o_g$ | the number of functional output terminals of the module $g$. Page 77. |
| $b_g$ | the number of functional bidirectional terminals of the module $g$. Page 77. |
| $s_g$ | the number of scan chains of the module $g$. Page 77. |
| $l_{g,k}$ | the scan length of the scan $k$ of the module $g$. Page 77. |
| $f_g$ | the number of scan flip-flops of the module $g$. Page 77. |
| $w_{max}$ | the maximal number of test wires assigned for the test scheduling. Page 77. |
| $c$ | the physical channel width of the NoC. Page 77. |
| $r()$ | the routing algorithm used by the NoC. Page 77. |
| vTAM | the tuple $\{d, k, w, \mathtt{R}_{atei}, \mathtt{R}_{part}\}$. Page 77. |
| $d$ | the FIFO depth assigned to the ATE interfaces and wrappers in this vTAM. Page 77. |
| $k$ | the packet size used to send test stimuli. Page 77. |
| $w$ | the number of test wires assigned to the ATE interfaces and wrappers in this vTAM. Page 77. |
| $\mathtt{R}_{atei}$ | the set of ATE interfaces in a given vTAM. Page 77. |
| $\mathtt{R}_{part}$ | the set of routers in a given vTAM. Page 77. |
| ATE interface | tuple $\{tw, d, \mathtt{V}\}$. Page 105. |
| $p_i$ | the parallel-to-serial loading and serial-to-parallel unloading time of the stimuli path. Page 86. |
| $p_o$ | the parallel-to-serial loading and serial-to-parallel unloading time of the responses path. Page 86. |

$c_i$ the number of data bits that comes from the NoC to the DfT module. Page 86.

$c_o$ the number of data bits that goes from the DfT module to the NoC. Typically $c_i = c_o = c$. Page 86.

SDI the set of selected data input terminals of an input test port. Page 88.

SDO the set of selected data output terminals of an output test port. Page 88.

RSDI the set of remaining selected data input terminals of an input test port. Page 88.

RSDO the set of remaining selected data output terminals of an output test port. Page 88.

DI the set of data input terminals of a test port. Page 88.

DO the set of data output terminals of a test port. Page 88.

CI the set of control inputs of a test port. Page 88.

CO the set of control outputs of a test port. Page 88.

$\mathtt{DI}_{in}$ the set of data input terminals of the input test port. Page 88.

$\mathtt{DO}_{in}$ the set of data output terminals of the input test port. Page 88.

$\mathtt{CI}_{in}$ the set of control inputs of the input test port. Page 88.

$\mathtt{CO}_{in}$ the set of control outputs of the input test port. Page 88.

$\mathtt{DI}_{out}$ the set of data input terminals of the output test port. Page 88.

$\mathtt{DO}_{ou}$ the set of data output terminals of the output test port. Page 88.

$\mathtt{CI}_{out}$ the set of control inputs of the output test port. Page 88.

$\mathtt{CO}_{out}$ the set of control outputs of the output test port. Page 88.

FI the set of all functional input terminals that are not part of the selected test ports. Page 88.

FO the set of all functional output terminals that are not part of the selected test ports. Page 88.

SI the set of scan-in terminals. Page 88.

SO the set of scan-out terminals. Page 88.

WC_SD1_COI the 'regular' wrapper cell. Page 89.

WC_SD1_COI_G the guarded wrapper cell. Page 89.

$s_i$ scan-in length. Page 91.

$s_o$ scan-out length. Page 91.

$T_{conv}$ conventional core test length. Page 91.

$t_i$ scan-in time. Page 92.

$t_o$ scan-out time. Page 92.

$T_{new}$ proposed core test length. Page 92.

$b_{test}$ maximal supported bandwidth of a given wrapper. Page 93.

$b_o^{out}$ maximal supported output bandwidth of the output port. Page 93.

$b_i^{in}$ maximal supported input bandwidth of the input port. Page 93.

$w^{max}$ maximal supported number of test wires of a given wrapper. Page 93.

$\mathtt{V}$ a set of the tuple $\{nburst, header, nword\}$. Page 105.

$nburst$ the number of test packets to send the test set. Page 105.

$header$ the header content of each test packet. Page 105.

$nword$ the number of flits of each test packet. Page 105.

$P_g$ the Pareto curve of a core $g$. Page 125.

# LIST OF FIGURES

# LIST OF TABLES

# LIST OF ALGORITHMS

# ABSTRACT

With the advance of microchip technology, global and long wires will cost more in terms of delay than in terms of logic gates. In addition, long wires are more susceptible to signal integrity problems such as crosstalk. A recently proposed global interconnect called network-on-chip alleviates the limitation of long wires. Moreover, on-chip networks allow decoupling communication and computation to divide a complete system into manageable and independent sub tasks. Thus, it is possible to integrate more logic into the chip using network-on-chip. However, the complexity growth of cores also increases the test costs since more logic is embedded into a single chip. These embedded cores need a test access mechanism for test data transportation, typically implemented as test-dedicated buses. As mentioned before, global wires are expensive, then, adding test buses may not be feasible in the near future. On the other hand, the on-chip network has access to most cores of the chip. This network could be used also for test data transportation, avoiding additional test-dedicated buses.

The goal of this thesis is to study the *reuse of on-chip networks for test data transportation*, looking for a *general reuse approach* that can be easily used in a given network. To reach this goal, the thesis is divided in three parts: models, design, and optimization. This thesis proposes a *functional model of a network*, compatible with most recently proposed best-effort on-chip networks. Based on this functional model, a *test model* is devised. The test model comprises of a set of necessary and sufficient information required to optimize the test architecture. The test architecture consists of DfT logic and scheduling algorithm. The *design of DfT logic* comprises adaptation logic for the external tester and test wrappers for the modules. The *optimization procedure*, focused on mesh-based best-effort NoCs, schedules test data such that the chip test length and DfT silicon area are minimized.

A conventional SoC test architecture based on test-dedicated buses is compared to the proposed approach for best-effort NoCs. The experimental results show that SoC test length has increased 5% on average. The results have also shown that the area overhead for proposed DfT is around +20% compared to the silicon area to implement the DfT of a conventional test architecture. On the other hand, we have also presented a simpler design flow and 20% to 50% of global wiring savings due to the use of NoC for test data transportation. The results corroborate with the conclusion that the proposed NoC reuse is a good approach for complex systems based on a large number of cores and routers.

**Keywords:** Modular Testing, System-on-Chip Testing, Core-Based Testing, Test Wrapper, Test Scheduling, Networks-on-Chip.

**Lógica e Escalonamento de Teste para Sistemas com Redes Intra-Chip Baseadas em Topologia de Malha**

# RESUMO

Com o avanço da tecnologia de fabricação de chips o atraso em fios globais será maior que o atraso em portas lógicas. Além disso, fios globais longos são mais suscetíveis a problemas de integridade como crosstalk. Uma proposta recente de interconnecção global chamada redes intra-chip reduz essas limitações referentes a fios longos. Além dessas vantagens, redes intra-chip permitem desacoplar comunicação e computação, dividindo um sistema em sub tarefas independentes. Devido as essas vantagens é possível integrar mais lógica em um chip que usa redes intra-chip. Entretanto, o acréscimo de lógica no chip aumenta o custo de teste. Os módulos do chip precisam de mecanismos para transportar dados de teste, que são tipicamente barramentos usados exclusivamente para teste. Entretanto, como mencionado anteriormente, fios globais são caros e acrescentar barramentos de teste pode não ser possível em um futuro próximo. Por outro lado, uma rede intra-chip tem acesso a maioria dos módulos do chip. Esta rede pode ser usada para transportar dados de teste, evitando o acréscimo de barramentos dedicados ao teste.

O objetivo dessa tese é estudar o uso de redes intra-chip para o transporte de dados de teste, enfatizando uma abordagem genérica que possa ser aplicada a uma dada rede. Para tanto, essa tese foi divida em três partes: modelos, projeto, e otimização. A tese propõe um modelo funcional de rede que é compatível com a maioria das recém propostas redes intra-chip. O modelo de teste, baseado no modelo funcional da rede, compreende o conjunto de informações necessárias para otimizar a arquitetura de teste. A arquitetura de teste, por sua vez, consiste de lógica para teste e algoritmo de otimização. A lógica de teste compreende lógica para ATE interface e lógica envoltória para módulos de hardware. Os algoritmos otimizam o tempo de teste e a área de lógica de teste no nível dos módulos e no nível do chip.

Uma arquitetura convencional de teste de SoCs baseada em barramento de teste dedicado foi comparada com a arquitetura proposta para SoCs baseados em redes intra-chip. Os resultados apontam que o tempo de teste do SoC com a arquitetura proposta aumenta em média 5%. Os resultados também mostram que a lógica de teste da arquitetura proposta é cerca de 20% maior que na arquitetura de teste convencional. Por outro lado, o fluxo de projeto baseado na arquitetura de teste proposta é mais simples que a convencional. Além disso, a arquitetura proposta reduz o número de fios globais em torno de 20% a 50% para SoCs complexos. Estes resultados demonstram que a arquitetura proposta é melhor para sistemas complexos com um grande número de módulos.

**Palavras-chave:** teste de sistemas intra-chip, lógica envoltória de teste, escalonamento de teste, redes intra-chip.

# 1 INTRODUCTION

The scaling of micro chip technology enables more logic, or an entire system, embedded in a single chip (System-on-Chip - SoC). It creates the opportunity to design tightly coupled parallel applications for example, for embedded systems in a portable device. However, it also brings challenges in terms of design productivity, design of global interconnect, and test for manufacturing defects (KEUTZER et al., 2000; ZORIAN; MARINISSEN; DEY, 1998).

The exponential shrink of the transistor size increases the available resources in a chip and increases the design complexity since more modules are embedded in the chip. In addition, the market competition demands a shorter design cycle. This motivates the adoption of some practices to deal with the design complexity, such as, core reuse, design partition, decoupling communication and computation, and higher levels of abstraction. The system is partitioned in independent sub tasks to ease its design and verification. These independent tasks rely on decoupling communication and computation to avoid the interference of other tasks. With independent tasks, higher levels of abstraction can be used to help the design. Moreover, the tasks are designed with standard interfaces to ease their integration. The system usually follows a layered design methodology to abstract the global interconnect implementation.

Another challenge of SoC design is related to the design of *global interconnect.* With the scaling of microchip technology, the computation is becoming cheaper than the communication because wires do not scale as transistor due to physical limitation of global wires such as signal integrity (fabrication defects, crosstalk, noise sensitivity) and power consumption. Figure 1.1 from ITRS illustrates the delay gap between wires and transistors for near future technologies. It can be observed that global wires are the most critical in terms of delay, thus, the time spent on global communication can overcome the time spent on local processing. For this reason, there is a *need for cost-effective global communication architecture* for future SoC design.

As an answer for the need for a cost-effective global communication for chips, one can observe that the functional interconnects have evolved from point-to-point links, to single and multiple hierarchical buses, and recently to networks-on-chip (NoC) (Figure 1.2). Each of these functional interconnects provides different features as illustrated in Figure 1.3. The increase of number of communicating cores suggests *shared* and *segmented* global wires to deal with, respectively, *routing area* and *long wires*. On-chip interconnect based on shared global wires are easier to *generalize*, reducing the design effort, since every core in the system just need to be connected to these global shared wires. However, shared global wires reduce the *bandwidth*

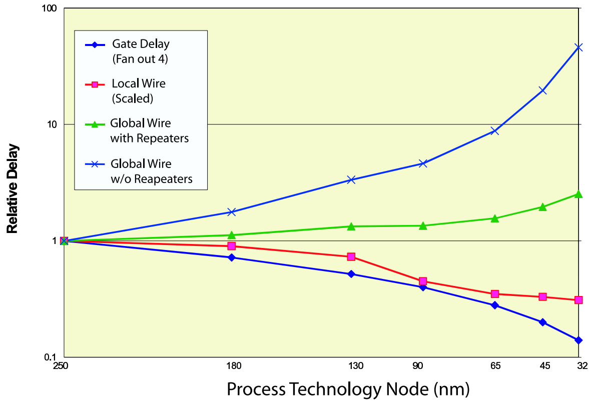


Figure 1.1: Projected relative delay for local wires, global wires, and logic gates for near future technologies from (ITRS, 2007).

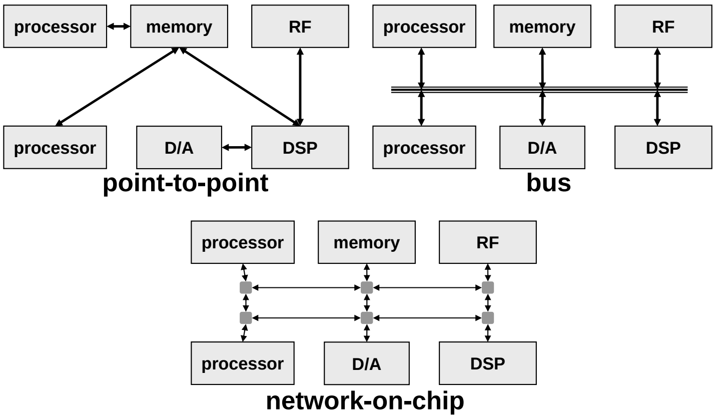


Figure 1.2: Evolution of communication architectures; from dedicated point-to-point links to buses and network-on-chip (adapted from (BJERREGAARD; MAHADEVAN, 2006)).

and increase the communication *bottleneck*, reducing the parallelism. In addition, segmented global wires increase the communication *latency*, but this problem can be reduced supporting pipeline.

Point-to-point links are optimum in terms of bandwidth, as they are designed for a specific application. But the number of links increases exponentially as the number of cores increases creating routing problems. Moreover, long wires are required in case of distant cores. Long wires degrade the signal, which become more sensitive to noise and crosstalk. *Shared global wires* alleviate the routing problem, but in case of buses, the problem related to long wires remains. Networks are an approach that share global wires but they are also *segmented* to avoid long wires. The evolution toward NoCs is a response for the need to reduce the design effort of complex applications (layered design approach, decoupling of communication from computation), need for generalized interconnect solutions, need to deal with Deep SubMicron (DSM) interconnect problems such as power, routing, performance, reliability, and predictability (DALLY; TOWLES, 2001; GUERRIER; GREINER, 2000; BENINI; DE MICHELI, 2002; JANTSCH; TENHUNEN, 2003; BJERREGAARD; MAHADEVAN, 2006). Table 1.1 has been adapted from (GUERRIER; GREINER, 2000; BJERREGAARD; MAHADEVAN, 2006)) to demonstrate some differences of

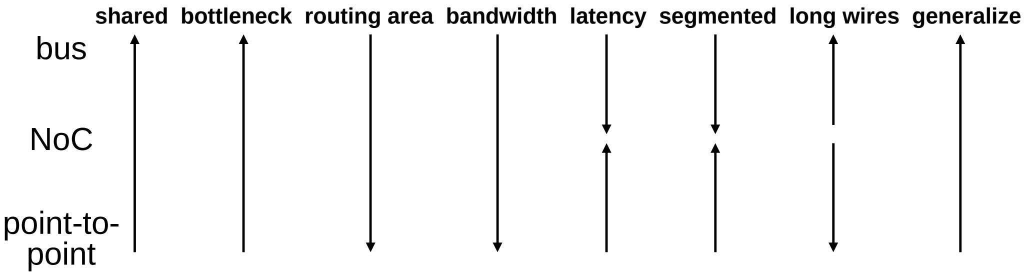


Figure 1.3: Comparison between communication architectures.

buses and networks.

Finally, although there are many practical issues to be addressed, it is generally agreed that the NoC approach offers several outstanding benefits for future SoCs (BENINI; DE MICHELI, 2002):

- Modularity, thanks to the ability to use basic components such as the Network Interface and the Router (to be presented in Section 2.2);
- Abstraction, a property of the layered approach;
- Flexibility/scalability of the network, as a consequence of packet-based communication;
- Regular and predictable electrical properties to cope with DSM issues;
- Re-use of the communication infrastructure viewed as a platform.

Testing SoC for manufacturing defects is an important challenge since it involves more logic to be tested (ZORIAN; MARINISSEN; DEY, 1998). The test of systems on board is based on the integration of previously tested ICs, and the "system" test comprises only the test of the interconnection among these ICs. However, the test of SoCs require testing all the modules. The increase of the number of transistors in a chip increases the challenges to test the chip. The main reason is the difficulty to access embedded logic through the pins.

NoC solves the problems related to global interconnect, which enables use of newer technology which are more susceptible to manufacturing defects and delay faults. The inclusion of delay test patterns increases the test volume. In addition, since the main motivation of NoC is to give support to design of more complex systems (i.e. more logic), it is expected more test data volume to be transported during test application, increasing the test time and test costs. On one hand, as stated before, the wires are becoming more expensive than logic. The test buses to access the embedded cores will be more expensive. On the other hand, the NoC has access to most cores of the chip and it also supports parallel communication. The *reuse of NoC to transport test data*, which is the focus of this thesis, seems to be an appealing approach to overcome the costs of test-specific buses. Cota et al. (2003) initially proposed the reuse of a *specific* NoC architecture for test data transportation. Compared to previous work, our approach is an attempt to *generalize* and *characterize the DfT costs* for a given network.

One could ask why it is relevant to study the reuse of NoC instead of point-to-point or buses. The first motivation is to *study communication architecture for future complex SoCs*, which has been demonstrated that both point-to-point and buses can not be applied efficiently. Indeed, there are papers that investigated the reuse of these interconnects for test (COTA et al., 2002; HUANG et al., 2001a; HARROD,

Table 1.1: Comparison between bus and network (adapted from (GUERRIER; GREINER, 2000; BJERREGAARD; MAHADEVAN, 2006)).

| bus | | | network |
|---|---|---|---|
| Every unit attached adds parasitic capacitance, degrading the electrical performance. | - | + | Only point-to-point one-way wires are used, therefore wire performance is not degraded. |
| Bus timing is difficult in a deep submicron process. | - | + | Network supports pipelined communication |
| Bus testability is problematic and slow. | - | + | Network can be used to transport test data and network supports multiple parallel communication |
| Bus arbiter delay grows with the number of masters. | - | + | Routing and arbitration logic are distributed. |
| The arbiter is instance-specific. | - | + | The same arbitration logic is implemented in each router |
| Bandwidth is limited and shared by all units attached. | - | + | Aggregated bandwidth scales with the network size. |
| Bus latency is zero once arbiter has granted control. | + | - | Internal network contention increase the latency (1). |
| The silicon cost of a bus is near zero. | + | - | The network has a significant silicon area (2). |
| Any bus is almost directly compatible with most available IPs, including software running on CPUs. | + | - | wrapper and conversion logic in both software and hardware are required (3). |
| The concepts are simple and well understood. | + | - | System designers need reeducation for new concepts (4). |

1999; BURDASS et al., 2000; HWANG; ABRAHAM, 2001; FEIGE et al., 1998). However, these approaches would not perform well on both test and functional domains in the near future. Moreover, a NoC is easier to *generalize* than point-to-point links, essential issue for any DfT approach; and it supports more *parallelism and bandwidth* than buses, essential to reduce the chip test time.

## 1.1 Problems to be Solved

We believe that the effective adoption of NoC reuse in actual designs depends on the following items:

- *The actual benefits of NoC reuse compared to conventional approaches based on dedicated TAMs*:
  The benefits and cost related to NoC reuse approaches are not clear enough. The most common claim of authors proposing NoC reuse is that it would save area since no test-specific TAM is required. On one hand, there are some design approaches based on dedicated TAMs that use information of placement of the cores to minimize the TAM wiring length, reporting negligible TAM costs (despite of the more complex design flow) (GOEL; MARINISSEN, 2003b). On the other hand, most papers about NoC reuse are about optimization algorithms under different constraints, but they do not focus on the requirements for the NoC reuse. i.e. which modifications are required in the design in order to reuse the NoC to transport test data ? What is the impact of the required DfT for NoC reuse in silicon area and test length compared to dedicated TAMs ? If the only benefit of NoC is to save wiring area, how much area are we saving with NoC reuse ?

- *Simple, general, and application-independent test approach*:
  Test designers have adopted modular testing approach for the test of complex chips since it is simple. It is mostly based on the well-know scan chains for intra-core access and simple TAMs, which are just wires, for inter-core access. Differentiated core structures, like memories, use BIST approaches. However, the BIST control itself is connected to the chip test control logic by, for example, boundary scan (i.e. more scan-based approaches). Simple approaches usually cost less in area and are easier to automate. In addition, scan can be used to most logic cores, so they are general and independent of design. On the other hand, NoCs are much more complex than conventional dedicated TAM, there is no general NoC design, and the NoC design is totally dependent on the application requirements; so, why reuse NoC for test instead of test-specific TAMs ? Is it possible to generalize NoC reuse ? Is it possible to simplify the *test view* of a given NoC ?

## 1.2 Goals

The strategic goal of this thesis is to propose a *general*[1] *approach for the reuse of on-chip networks for test data transportation.* To accomplish this strategic goal, the following specific objectives should be fulfilled:

[1] The term 'general', unless specified, refers to a test approach that is applicable to most best-effort and mesh-based NoC designs, which was the class of NoC most used along this thesis.

- Make the requirements for NoC reuse explicit;
- Determine the test logic required to enable the reuse of both best-effort (BE) and guaranteed throughput (GT) NoCs;
- Determine the optimization algorithms for the DfT modules;
- Propose a test schedule tool for overall test architecture optimization considering BE NoCs;
- Compare the proposed test architecture with the conventional test architecture based on dedicated TAM to establish the actual advantages and drawbacks of NoC reuse.

## 1.3 Contribution

The main contribution of this thesis is a general test model for NoC-based SoCs such that this proposed model is compatible with the current test methods. Others contributions are:

1. Concise functional NoC definition (Chapter 1);
2. Extensive analysis of prior work (Chapter 3);
3. General test model for a BE NoC-based chip (Chapter 4), partially published in Amory (2007);
4. Detailed design and optimization of DfT required to enable the reuse of both BE and GT NoCs as TAM (Chapters 4 to 7), published in Amory et al. (2005; 2006; 2007; 2007);
5. Test planing tool based on the proposed test model to optimize the chip test length (Chapter 8), to be published.

## 1.4 Outline of the Thesis

This thesis is organized in three parts as presented below.

### 1.4.1 Background, Definitions, and Models

Chapter 2 presents background related to SoC testing and NoCs. In sequence, the prior work is reviewed in Chapter 3; topics like wrapper design, TAM design, FIFO testing, and recent papers about test of NoC are analyzed. Later, in Chapter 4, *we propose a test model that represents the necessary and sufficient information of a BE NoC-based system to implement the DfT and optimization procedures proposed in this thesis.*

### 1.4.2 DfT Design for NoC Reuse

Once the NoC definition is established and the set of required information for test is defined, we detail the design processes required to modify the chip for test. The second part of this thesis, in Chapters 4 to 7, presents the *DfT logic required to use both BE and GT networks to transport test data.*

To use the network for test, some additional logic is required to interface the external tester to the network, to the cores, and to the network building blocks as well. It has been identified, for example, that not only the cores, but also the test pins require a wrapper logic in order to connect to the NoC. Besides, the wrapper requires a different design compared to the conventional modular test approaches because the "TAM" works with well-defined protocols. The proposed wrappers have been designed to abstract the test path as "pipelines" that can pump data at different rates.

### 1.4.3 Test Optimization

Considering the chip information required for test enumerated in the test model and the required test circuitry, the final step is to integrate it into a test planning tool that optimizes the test architecture in terms of test length and silicon area (Chapters 8 and 9).

# 2 RELATED BACKGROUND

The following sections present basic concepts related to test of SoC, basic concepts of NoC design, and the outline of this thesis.

## 2.1 Modular Testing

*Modular testing* (GOEL; MARINISSEN, 2003a), i.e. testing individual SoC modules as stand-alone units, has been used for SoCs. *Non-logic modules* such as embedded analog and memories require modular testing since they use different test strategies then random digital logic. The ability to use the most appropriate test strategy for every core increases the *test quality*. *Black-boxed cores*, such as hard or encrypted cores which no implementation detail is given, require modular testing because they need to be tested with test vectors supplied by the core provider. In addition, modular testing provides an attractive "divide-and-conquer" test development approach that reduces the time to generate the test vectors, and allows for *test reuse* when a core is used in multiple designs.

Nevertheless, a core is typically embedded in the SoC and there is no direct access from the SoC pins to the core terminals. Modular testing requires every core to have *test access* from its terminals to the chip pins, and to be *isolated* from its surrounding circuitry. Zorian et al. (1998) introduced a general conceptual test architecture (Figure 2.1) for modular testing of SoCs. It consists of three elements per Core-Under-Test (CUT): (*i*) a test pattern *source* and *sink*, which generates and evaluates the test stimuli and responses, respectively (*ii*) a *Test Access Mechanism* (TAM), which transports test data from the source to the CUT and from the CUT to the sink, and (*iii*) a *test wrapper* for isolation during test mode and to enable switching between the functional access to the test access through the TAM.

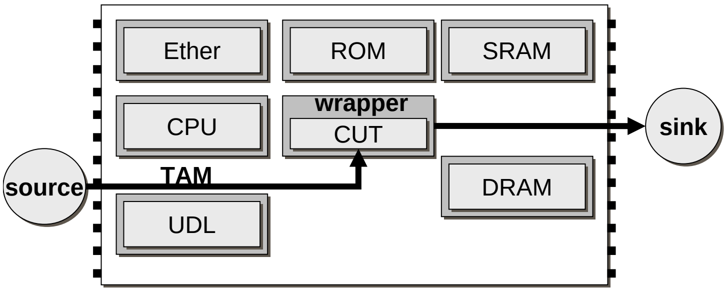


Figure 2.1: general conceptual test architecture (from (ZORIAN; MARINISSEN; DEY, 1998)).

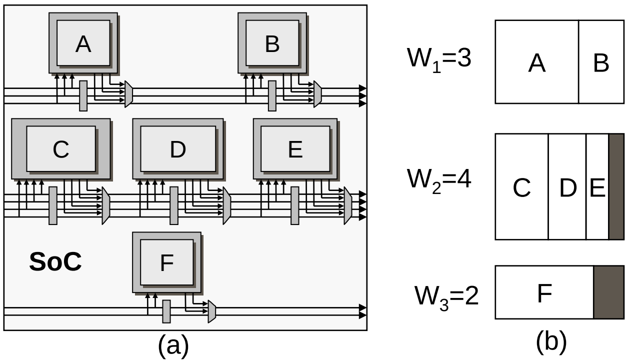


Figure 2.2: (a) Example of TestRail architecture and (b) corresponding test schedule (adapted from (GOEL; MARINISSEN, 2003a)).

The *test architecture* design problem can be defined as (GOEL; MARINISSEN, 2003a): for a given *set of cores* and a given *number of test pins*, determine the *TAM type*, the *number of TAMs*, the *width of each TAM*, the *assignment of cores to the TAMs*, and the *wrapper design*, such that the *chip test length* is minimized. Several papers have investigated test architecture designs, such as test bus (VARMA; BHATIA, 1998) and TestRail (MARINISSEN et al., 1998), and how to optimize the test architecture (CHAKRABARTY, 2000; MARINISSEN; GOEL; LOUSBERG, 2000; HUANG et al., 2001b; GOEL; MARINISSEN, 2003a). Figure 2.2 illustrates an example of resulting test architecture and test scheduling (GOEL; MARINISSEN, 2003a).

Two test flows are required to test a core. The *stimuli test flow* transports the test stimuli from the source to the CUT and the *response test flow* transports the test responses from the CUT to the sink. The stimuli and the response test flows may transport different data amounts. For instance, let us assume the test of a core with one input terminal, no internal scan chains, and one hundred output terminals. Each test pattern of this core requires one bit in and one hundred bits out. The *test data volume*, the number of bits required to test a core, considering both stimuli and responses, is given by the core provider, thus, the test data volume is considered invariant.

Since the amount of test data is fixed, the core test length[1] depends on how fast the amount of test data can be transported over the TAM. This variable is called *test data rate*. A recommended test data rate should range from 1 bit/clock cycle to a certain maximum which depends on the core and on the TAM width. The minimal test data rate of 1 bit/clock cycle is recommended, but not mandatory, because it keeps a minimal but constant data flow and it reduces the core test length.

Interruptions on the data flow require mechanisms to halt the test. The usual approaches are the use of clock gating or holdable scan cells. Clock gating halts the clock for a certain period. The drawback is that it requires changing the clock tree, which is generally not recommended for complex systems. The other approach to freeze the test is to implement holdable scan cells, i.e. scan cells that can keep the current value, but this approach requires an extra multiplexer for each scan cell. Holdable scan cells increase the area for DfT. Moreover, it is not feasible for

[1] the term test length is used when the unity is clock cycles. Test time is used when we refer to seconds.

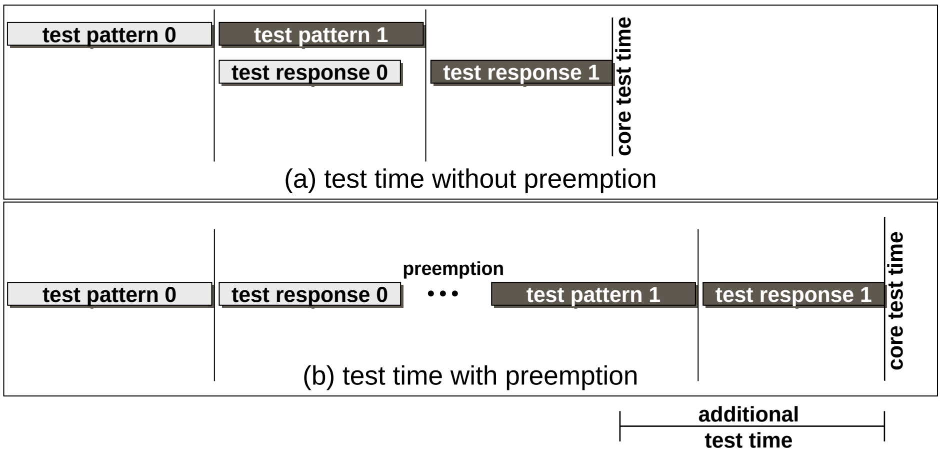


Figure 2.3: Core test length for non-preemptive and preemptive test.

hard-core since it is not possible to change the hardware description.

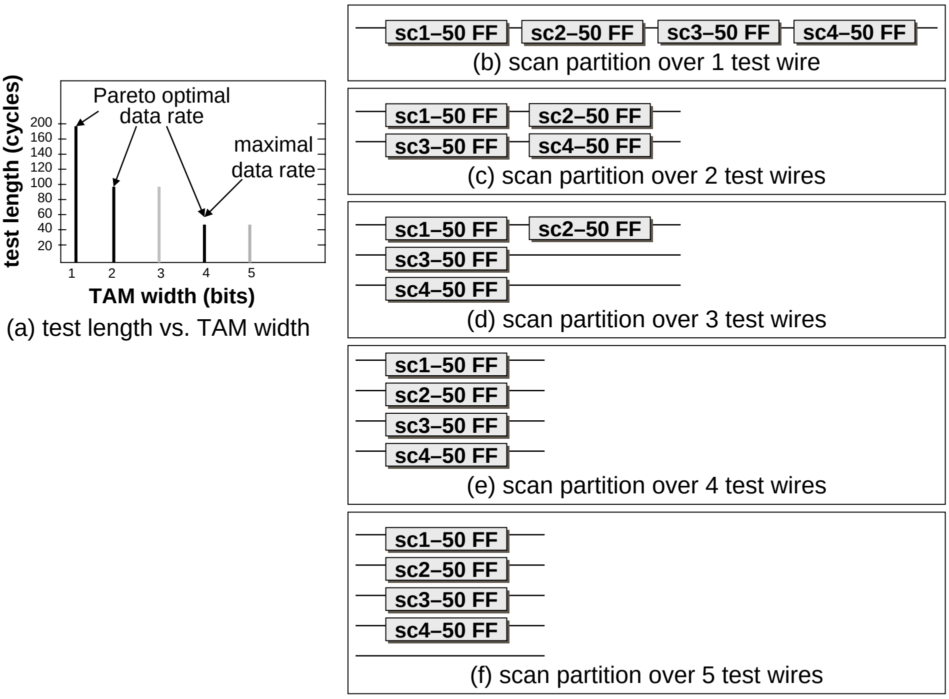
Interruptions in the data flow have an additional drawback in test length. Every time the flow is interrupted, the parallelism between scan-in and scan-out is broken, increasing the core test length. This effect, illustrated in Figure 2.3(a), is referred as *test pipeline*. Figure 2.3 shows that preemptive test increases the core test length since scan-in is not carried out in parallel with the scan-out.

Another drawback related to interruptions or gaps in the data flow is that most ATEs expect test stimuli and responses as streaming data. ATEs are typically not prepared for more complex interactions with the SoC.

On one hand, assuming a constant test data rate avoids halting the test. On the other hand, the TAM has to guarantee this constant data rate. Dedicated test architectures like the one illustrated in Figure 2.2 naturally guarantee a certain test data rate because the TAM is basically just wires and buffers connecting the test pins to the CUTs. In addition, each core in a TAM is usually tested sequentially to guarantee the exclusive access to the core.

Figure 2.4 illustrates the effect of the assigned test wire on core test length. It also demonstrates two concepts: *Pareto optimal test wires* (a multi-variable optimization) and the *maximal test wire* (GOEL; MARINISSEN, 2003a). For example, let us assume a hypothetical hard-core with four internal scan chains, each one with 50 flip-flops. Assigning just one test bit for the test of this core results in a scan length of 200 cycles because the four scan chains are concatenated. As more test wires are assigned, the scan length reduces. However, the reduction has a limit, in this case, when 4 bits/cycle are assigned. Beyond this limit the scan length does not reduce. Moreover, the scan length reduction plot does not have a perfect stair-case behavior. Some situations, like the one with 3 bits/clock cycle, may not result in a reduction in the scan length compared to, for example, 2 bits/clock cycle. This effect occurs due to the sizes of the internal scan chains.

Only scan configurations that strictly reduce the scan length are allowed. That is, if two scan configurations lead to the same scan length for a core, the one with smaller number of test wires is chosen. In our example, 4 is a Pareto optimal test wire because 1, 2, and 3 lead to a higher test length than 4. In contrast, 5 is not Pareto optimal because 4 leads to the same test length than 5. Maximal test wire is the maximal Pareto optimal test wire. The maximal bandwidth for the example in



(a) test length vs. TAM width

(b) scan partition over 1 test wire

(c) scan partition over 2 test wires

(d) scan partition over 3 test wires

(e) scan partition over 4 test wires

(f) scan partition over 5 test wires

Figure 2.4: Core test length vs. bandwidth.

Figure 2.4 is 4 bits/cycle. Defining the test bandwidth to the maximum would minimize the core test length. Nevertheless, it would require expensive communication resources from the chip. Moreover, a SoC typically has multiple cores. Assigning the maximal bandwidth for each core is unfeasible since the number of test pins is usually small. One of the goals of a test scheduling tool is to find this tradeoff such that the chip test time is minimized. A cost effective distribution of test bandwidth to the cores to minimize the test length is a common subject of research (CHAKRABARTY, 2000; HUANG et al., 2001b; IYENGAR; CHAKRABARTY; MARINISSEN, 2002a; GOEL; MARINISSEN, 2003a).

## 2.2 Networks-on-Chip

This section presents basic concepts regarding NoCs. It is not intended to be a extended review of network design; for such, titles like Duato et al. (2003), McCabe (2003), Jantsch and Tenhunen (2003), and Bjerregaard and Mahadevan (2006) can be used. The emphasis is given on the basic concepts, the most important building blocks (architecture and hardware implementation), their role (capabilities that can be useful for test), and their interaction (protocols, timing, and terminal-to-terminal binding).

A given application in the context of chips consists of nodes, which are typically processing units, memory, standard I/O, or combination of them, that need to communicate with each other to complete a certain task. Different kinds of communication media can be employed; each of them with different costs and performance. NoCs are a recent proposal which are more cost effective for very complex designs. In a NoC-based chip, cores communicate with each other via a network which consists of *network interfaces* (NI), *routers*, and links (two *channels* in opposite directions) (RADULESCU et al., 2005). Figure 2.5 illustrates a floorplan and a common logical

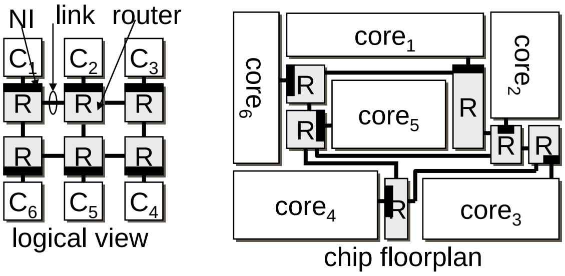


Figure 2.5: A simple NoC instance.

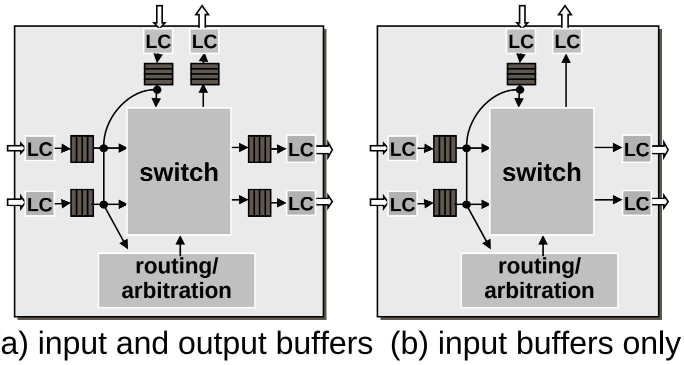


Figure 2.6: A typical router architecture (DUATO; YALAMANCHILI; NI, 2003). LC denotes link controller.

view of a possible NoC instance.

A router, which can also be called switch in the literature, is the main building block of a network; it handles the message communication among the nodes. A router has a number of input and output ports (i.e. the router *degree*), where one of these ports may connect to a node and the remaining connect to neighbors or adjacent routers. The way that adjacent routers are connected define the *topology* and can be modeled by a graph $G(N, C)$, where the vertices of the graph $N$ represent the set of routers and the edges of the graph $C$ represent the set of channels. Many topologies have been proposed to balance performance and cost parameters.

Nodes communicate with each other sending *messages*, which for performance reasons may be divided into *packets* before the transmission. Packets are the unit of communication that contains the packet *header* which carries additional information such as the destination address. A *routing algorithm* determines the intermediate routers transversed (the path) by the packets to reach the destination. The routing algorithm determines the output port of a packet.

A typical router architecture is illustrated in Figure 2.6. Inside a router, when a packet header arrives in an input channel, the *switching* mechanism determines how and when the network resources are allocated for the message transfer (DUATO; YALAMANCHILI; NI, 2003). The typical resources are channels and *buffers*. The buffers are used to store data temporally until the output channel is chosen or it is freed. *Flow control* mechanism establishes the communication between two adjacent routers, controlling the data flow as the buffer space is free. Examples of channel flow control are illustrated in Figure 2.7.

The architecture of a router, illustrated in Figure 2.6, typically consists of the following components (DUATO; YALAMANCHILI; NI, 2003):

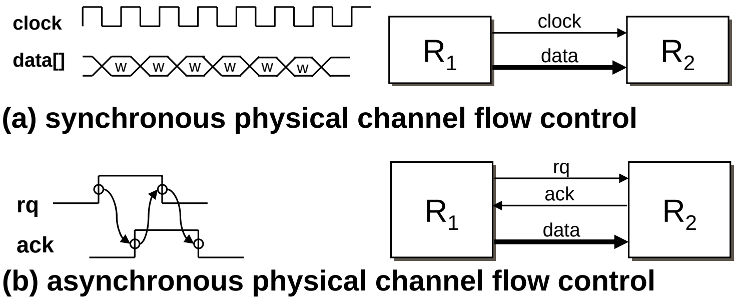

clock
data[]
w
R1
R2
clock
data
(a) synchronous physical channel flow control
rq
ack
R1
R2
rq
ack
data
(b) asynchronous physical channel flow control

Figure 2.7: Examples of physical channel flow control (adapted from (DUATO; YALAMANCHILI; NI, 2003)). (a) synchronous and (b) asynchronous.

- *Buffers*: They are First-In First-Out (FIFO) buffers used to store messages in transit. They can be implemented in the input ports (Figure 2.6(b)), output ports, or both (Figure 2.6(a)).
- *Switch*: This component is responsible for connecting the input ports to the output ports.
- *Routing and Arbitration*: This component implements the routing algorithm and selects the output port for an incoming message. This component also provides arbitration in case there are multiple requests for the same output port.
- *Link Controller*(LC): This component implements the channel flow control between adjacent routers.

The switching mechanism, flow control mechanism, and buffer management have direct impact on the network performance (DUATO; YALAMANCHILI; NI, 2003; MCCABE, 2003). Let us define some performance metrics. *Throughput* is the rate (bits/s) at which a network device sends or receives data. *Bandwidth* is the maximum theoretical throughput. It indicates the maximum amount of data that can pass from one point to another in a unit of time. Throughput is more like a measure of the usage of a resource and bandwidth is the measure of the resource itself. *Latency*, or delay, is the time elapsed from the beginning of data transmission until the time data is received at destination. *Jitter* is a measure of delay variation over time.

Routers with different designs and architecture require a different analytical performance model. For example, Equations 2.1 and 2.2 model the no-load packet latency (latency in the absence of any traffic) for the router architectures depicted in Figure 2.6(a) and (b), respectively (DUATO; YALAMANCHILI; NI, 2003).

The packet to be transferred has a size of $L$ bits. The physical data channel width has $W$ bits. Assuming a one word header, the total packet size is $L/W$ bits. The router does the routing decision in $t_r$ seconds; the physical channel runs at $B$ Hz. Thus, the physical channel bandwidth is $B \times W$ bits per second. Assuming that the channel wires can complete a transmission in one clock cycle. Therefore, the propagation delay is $t_w = \frac{1}{B}$. Once the routing path has been set up, the switching delay is denoted by $t_s$. Thus, a word can be transferred from the input channel to the output channel in $t_s$ seconds. The distance between the source and destination is assumed to be $D$ hops. Figure 2.8 illustrates these variables. The expression $D(t_r + t_s + t_w)$ represents the network latency, while the rest of the equation represents the time to transfer a given amount of data $L$ over $W$ wires

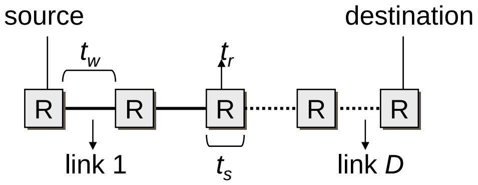


Figure 2.8: Latency from the source to the destination (adapted from (DUATO; YALAMANCHILI; NI, 2003)).

in a pipelined manner. With input and output buffers, the pipeline cycle time is determined by the maximum of the switch delay and the wire delay $\max(t_s, t_w)$. For input-only or output-only, the cycle time is the sum of $t_s + t_w$.

$$t_{wormhole1} = D(t_r + t_s + t_w) + \max(t_s, t_w) \times \left\lceil \frac{L}{W} \right\rceil \tag{2.1}$$

$$t_{wormhole2} = D(t_r + t_s + t_w) + (t_s + t_w) \times \left\lceil \frac{L}{W} \right\rceil \tag{2.2}$$

These equations exemplify how router design decisions impact the network performance. *This level of design details may be considered complex for our goal of designing general network models* since there is a huge number of possible design variations, thus, huge number of corresponding analytical models.

Up to now, just the most basic communication ‘services’ have been described. However, complex applications may require more complex services. For examples, some applications may require additional reliability, thus, services as error detection, error correction, and packet retransmission may be implemented. Choosing a proper boundary and location of the required services are perhaps one of the first design steps. The *end-to-end design principle* (SALTZER; REED; CLARK, 1984), applied for distributed systems, says that it is usually better to place application-dependent functions (or services) closer to the application that uses the function. In the case of core-based chip design the ‘application’ is a core and the ‘function’ is, for example, a mechanism to give support to reliable data transfer. For this reason, only the basic communication services are implemented in the routers (in side the network) and the optional and application-dependent services are implemented in the boundaries of the network with the cores, commonly called *Network Interface* (NI). Note that the application-dependent services could also be implemented in the cores, but it would impair the core reusability to other systems.

The services implemented in the NIs are usually described in a *layered* manner, with the advantage of abstracting the most basic services which are closer to the implementation details (MCCABE, 2003; RADULESCU et al., 2005; KEUTZER et al., 2000). The global on-chip communication can be decomposed into five layers (TANENBAUM, 1996). The protocol stack enables different services and allows Quality of Service (QoS) mechanisms, providing to the programmer an abstraction of the communication framework. Layers interact through well-defined interfaces and they hide the low-level physical DSM issues. Figure 2.9 correlates the most important protocol layers and where they are implemented in the NoC. The Physical layer refers to the electric details of wires, the circuits and techniques to drive

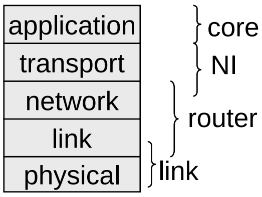


Figure 2.9: Protocol layers and the main building blocks of a NoC-based design.

information (drivers, repeaters, and layout). The Data Link level ensures a reliable transfer and deals with medium access (sharing/contention). The Network level deals with issues related to the topology and routing scheme. Finally, the Transport layer manages the end-to-end services and the packet segmentation/re-assembly. QoS, which helps to decouple communication from the computation, is also implemented at the transport layer. The Application is implemented in the set of cores of the chip. For the core point-of-view the services should be transparent (abstracted). For example, a core sends and receives raw data (without packetization) without knowing that there is a NI that provides reliable data transfer.

The communication services can be implemented either in software or in hardware, however, in the context of chips, which has tighter *latency constraints*, it is preferable to implement these services in hardware. In addition, recall that not all nodes in a NoC are programmable, thus, if additional services are required, they have to be implemented in hardware. Therefore, just an indispensable set of services is implemented since there is also *area constraint*. The set of most used network services are (MCCABE, 2003): *packetization*, responsible for inserting and removing the header, the sequence number, the parity, the CRC, or the check sum information into the packet; *buffering* used to overlap communication an computation, which requires an end-to-end flow control mechanism to control the buffer usage; *packet ordering* in case adaptative routing is supported, which also requires sequence number in the packet header; *reliable data delivery* ensures that the destination receives the correct information. It may include error detection, error correction, or packet retransmission mechanisms; *soft or hard performance guarantees* for latency, jitter, or bandwidth, which may require mechanisms for resource reservation; *collective communication* like the support of one-to-many and one-to-all communication patterns rather than just one-to-one.

Another relevant consideration is how the cores are connected to the network. It has been demonstrated that the cores are connected to the router network through a NI, however, NIs need a standard interface to do the terminal-to-terminal connection of the cores to the NoC. Thus, the NoCs assume that both cores and NIs are connected via a *memory-mapped on-chip port* like OCP (OCP-IP, 2003), AXI (ARM, 2004a), VCI (ALLIANCE, 2000), or DTL (PHILIPS SEMICONDUCTORS, 2002). The advantage is that no matter the NI design, the connection of the cores and the network is facilitated, reducing the design effort. Such practice has been used for other interconnect, like buses. Indeed, these memory-mapped types of ports are inherited from buses. They are used for NoCs just to keep the compatibility with existing cores, since they increase the packet latency when used for NoCs (OST et al., 2005). In the near future new communication approaches specific for networks (like message passing) may come up for NoCs to reduce the packet latency.

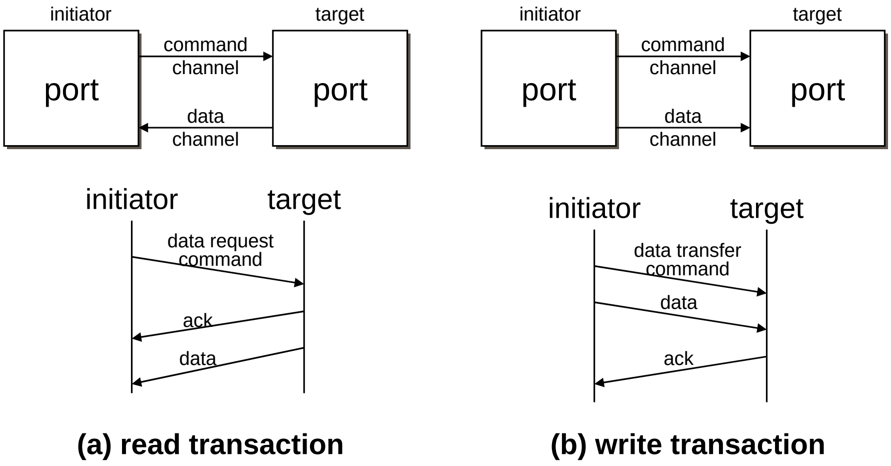


Figure 2.10: Initiator and target ports; read and write transactions.

A port may be classified as *initiator or target* (sometimes also called master or slave), and as *read, write, or read/write port*. An initiator port starts transactions which are sent to a target port; a target port just receives and executes transactions. A transaction can be split into read or write commands. A read command (Figure 2.10(a)) sends a request command from the initiator to the target port; the target port acknowledges it and sends the requested data. A write command (Figure 2.10(b)) sends data from the initiator to the target; the target consumes the received data and acknowledges it. A port is usually configurable, thus, it may not necessarily support both read and write commands. For example, some valid port usages are: an initiator read port starts read commands; an initiator write port starts write commands; a target read/write port answers both read and write commands.

Figure 2.11 illustrates an OCP-like port and its protocol. The main principles can be applied to other protocols. Event 1 in Figure 2.11 represents the request of a write command to send two words, which is accepted in the next clock cycle, during the event 2. The first word is sent during the event 3, when the data valid is high. In event 4 the target does not accept the second word, but it is accepted in event 5. The read command works in a similar way.

One may have realized that several concepts about router design were not presented on this Section. Recall that the main goal of this thesis is to conceive a *general* test approach for NoC based on network reuse to transport test data. If all kind of design decision is required to a model, the generality is compromised. Thus, just the required notions were presented.

### 2.2.1 Functional NoC Model

This section presents the informal functional NoC model and its assumptions. This is the functional NoC model used in this thesis. Figure 2.12 demonstrates one valid instance of the proposed functional model:

- *Routers* are connected to other routers and to zero or one NI via links;
- A *link* consists of two pairs of router ports and two channels such that data flows in both directions in the *channels* (see Section 2.2.1.1);
- *Router ports* implement the channel protocol, like the one illustrated in Fig-

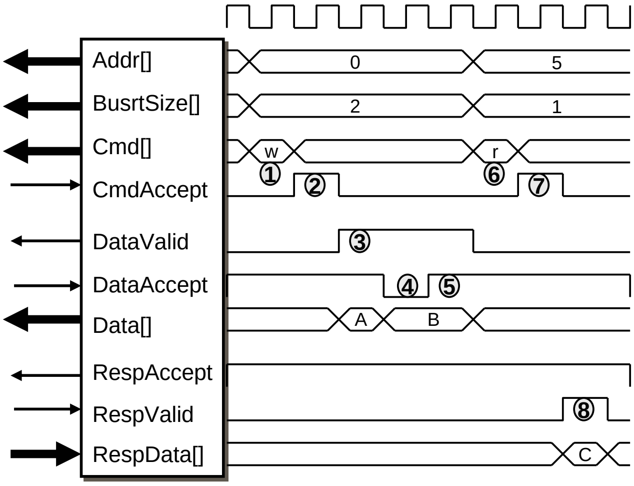


Figure 2.11: Protocol of an OCP-like port.

ure 2.7, and all router ports have the same configuration, i.e. the same width, bandwidth, and protocol (see Section 2.2.1.1);

- *Core ports* implement an on-chip protocol such as OCP. Each core port is configured individually and the data flow can be unidirectional or bidirectional;
- A *NI* has one or more core ports and only one router port;
- A *core* connected to the NoC has two or more core ports. Core ports are not required for cores not connected to the NoC;
- A NI connects to one or more cores via core ports;
- A core connects to zero or more NIs via core ports;
- *Pins* are connected only to cores (see Section 2.2.1.2);
- Only *synchronous* systems are considered (see Section 2.2.1.3);
- The routers must be organized in a *mesh topology.*

The model requires NIs to do the interface among the cores and the routers. The *minimum requirement for NI is packetization* (i.e. conversion from a core protocol like OCP to a channel protocol like handshake) to abstract the packet format from the cores. Thus, the cores manipulate raw data through the on-chip ports. Core are typically memory mapped, thus, the address information sent from the core to the NI via the port is used to select the destination. In this way, the cores do not realize the existence of a NoC; they just have to obey the on-chip protocol.

Note that there is no information related to the implementation of the modules, except by packetization for NIs and routing algorithm. For instance, we do not require multicast service, or any other kind of network service. For this reason this functional model is used as a stepping stone for the proposed general test model presented in Chapter 4.

#### *2.2.1.1 Link and Router Configuration*

There might be NoCs with unidirectional links, like in ring topologies. In additional, it is also possible to have NoCs whose channel widths are different. For instance, a certain channel might require higher bandwidth, so wider channel width might be used. Likewise, there might be routers whose ports have different width or even different channel protocols. We believe that these considerations might be implemented, however, they are unusual. Thus, we assume, for sake of simplicity,

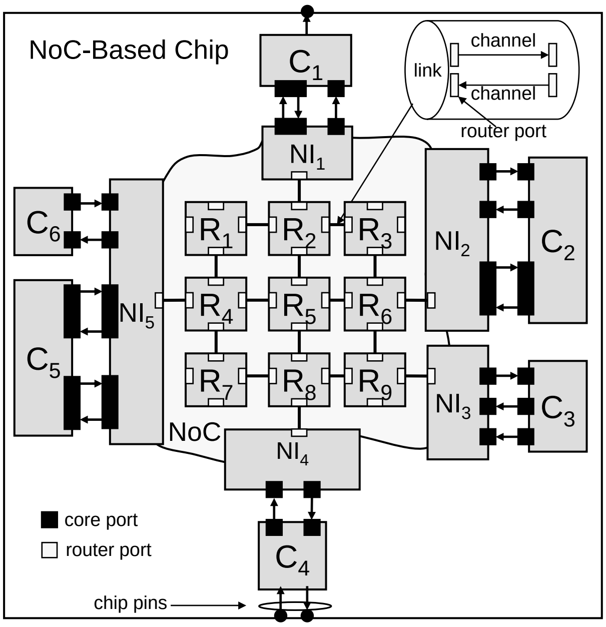


Figure 2.12: A more complex NoC instance.

that all channels have the same width, single channel protocol, and bidirectional links.

### *2.2.1.2 On-chip vs Off-Chip Protocols*

*We assume that pins are not connected directly to the NoC, i.e. there will always have some logic between the pins and the NoC.* The reason for this realistic assumption is that on-chip and off-chip communication requires different interfaces, buses, and protocols (the rest of the document uses just protocol). Examples of on-chip protocols are wishbone (OPENCORES, 2006a), OCP (OCP-IP, 2003), AMBA AXI (ARM, 2004a), AMBA APB (ARM, 2004b), VCI (ALLIANCE, 2000), DTL (PHILIPS SEMICONDUCTORS, 2002), CoreConnect (IBM, 2006) while some examples of off-chip protocols are PCI express (PCI-SIG, 2006), RapidIO (RAPIDIO, 2006), FireWire (IEEE, 2006), USB (USB, 2006), Rambus FlexIO (RAMBUS, 2006a), Rambus XDR (RAMBUS, 2006b), Rambus RDRAM (RAMBUS, 2006c). Another reason is, as stated in Section 2.1, most ATEs do not support more complex interaction with the SoC, like executing a protocol.

The complexity of the off-chip protocols is the main reason why they are not used for on-chip communication. On-chip protocols are simpler and require less silicon area while off-chip protocols are more complex. For example, a Firewire core (CAST, 2006a) requires 38,834 gates, a PCIe core (CAST, 2006b) requires from 35,500 to 50,700 gates, and a USB core requires 13,800 gates. On the other hand, on-chip interfaces require around 1,000 gates depending how it is configured. Off-chip protocols are more complex because they require extra logic for reliability issues (error control), high-performance signaling, complex synchronization schemes. Typical on-chip protocols do not require such features, thus, they are much simpler.

In the typical scenario an interface core is the logic connected to pins, not the NoC. Figure 2.13 shows a typical block diagram of a PCI-Express interface core from CAST (CAST, 2006b). The main core is the PCIe-EP module while the PCIe-IF

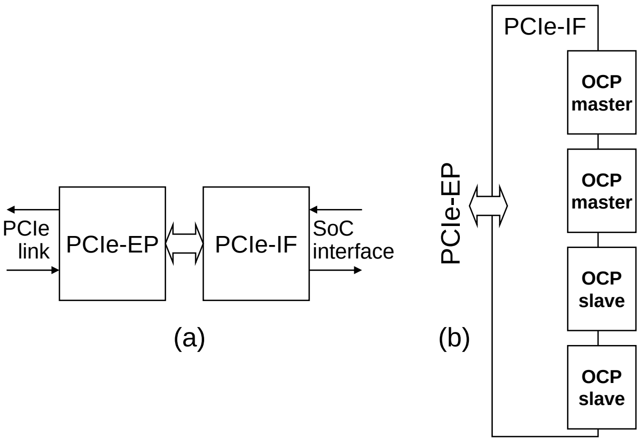


Figure 2.13: Block diagram of the PCI Express IP core (from (CAST, 2006b)).

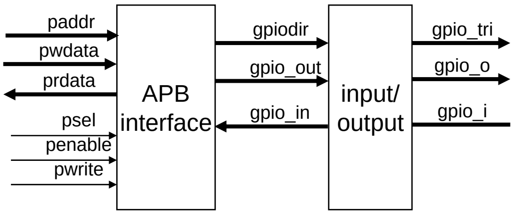


Figure 2.14: Block diagram of the GPIO IP core (from (CAST, 2006c)).

does the interface with OCP protocol. The module PCIe-IF has four independent OCP ports. This architecture where the main core is separated from the interface ease the support of other on-chip interfaces. For instance, this PCIe core also supports wishbone and AMBA APB interfaces. The same core provider also has a Firewire core (CAST, 2006a) with support to several interfaces. These examples of commercial IP cores show that the use of standard protocol for inter-core communication is a common practice.

Even when pins are not related to an off-chip protocol, some logic is still required to adapt the pins to the on-chip protocol. This is the case of the GPIO IP core (CAST, 2006c) which connects a configurable number of input and output terminals to AMBA APB protocol. Figure 2.14 illustrates the block diagram of this IP core. The right side shows the IO pins and the left side shows the AMBA APB interface.

These two examples illustrate that typically there will be some protocol conversion logic between the pins and the NI. Thus, a DfT logic is required to connect the test pins the NoC during test application. One might think that it could be possible to use the native protocol to send data from the pins to the NoC. The problem is that the ATE would need to use, for instance, USB, PCIe, or any other off-chip protocol, which is not realistic nowadays.

#### *2.2.1.3 Synchronous and Single-Clocked Systems*

*We assume the system is synchronous and has a single test clock frequency for sake of simplicity.*

Since there are multiple buffers in a NoC, they can be used to decouple different test frequencies. Liu et al.(2005) presented a high-level NoC model with support to reusing NoCs with different frequencies. However, there is no information about the modification of DfT modules to support this feature. Future work can detail the DfT for the support of multiple test frequencies.

There are also asynchronous networks (FELICIJAN; FURBER, 2004; BAINBRIDGE; FURBER, 2002; BJERREGAARD; SPARSO, 2005a; BJERREGAARD, 2005; WANG et al., 2005). The first papers about testing asynchronous networks came up recently (EFTHYMIOU; BAINBRIDGE; EDWARDS, 2004, 2005; TRAN et al., 2006). The DfT strategy for asynchronous circuits is slightly different from the one for synchronous circuits. For instance, asynchronous circuits have feedback loops that are hard to test. Scan cells are used to break these loops, however, a full-scan approach leads to a very high area overhead (typically 80%) (BEEST et al., 2002; EFTHYMIOU; BAINBRIDGE; EDWARDS, 2005). For these reasons it is not the scope of this work to support asynchronous networks.

### 2.2.2 Some Industrial NoC Approaches

This section briefly highlights some features of some industrial NoC instances. Analyzing industrial applications of NoC is not an easy task since details about tools and design are not publically available. However, it is an interesting exercise because it helps to visualize short to medium-term industrial use of NoCs. Finally, this section shows that the first test chips are coming up (CLERMIDY; VARREAU; LATTARD, 2005; LEE S-J. LEE; YOO, 2005) and that other companies are joining the research effort. Moraes et al. (2004) and Bjerregaard and Mahadevan (2006) surveyed several other NoC approaches.

Philips Research has investigated different facets of NoC design, like design of basic building blocks like router and network interface (RIJPKEMA et al., 2003; RADULESCU et al., 2005), and design of tools for rapid and cost-effective NoC instantiation like application mapping tools, NoC synthesis, NoC debug, NoC formal and functional verification (GOOSSENS et al., 2005; MURALI et al., 2006; CIORDA et al., 2006). The target application domain of the Æthereal NoC platform is embedded systems and consumer electronics with real-time performance constraints. Goossens et al. (2005) presented a communication centric design flow where the applications consist of a number of communicating tasks. These tasks are characterized in terms of communication requirements, like maximum latency, minimum bandwidth, traffic class (best-effort or guaranteed). Later, the application is mapped onto an application-specific NoC topology, and its VHDL code is generated and verified. The supported on-chip interfaces are OCP, DTL, and AXI.

ST Microelectronics (2005) presents the STNoC which implements the Spidergon topology, based on a regular chordal ring. Few informations are available about it, however, it is possible to compare some feature with the Æthereal approach; Æthereal is based on application-specific topology, while STNoC is based on regular and homogeneous router design with three ports. Both approaches support best-effort and guaranteed communication services. STNoC supports OCP on-chip interface (MAGARSHACK; PAULIN, 2003).

Silistix (2006) is a spin-off from the University of Manchester. The company's focus is the development of self-timed on-chip network architecture called CHAIN (BAINBRIDGE; FURBER, 2002). Tools are available for NoC instance with support to OCP and AXI on-chip interfaces. The target is low power application domain. Although there is no information about support to guaranteed services, the issue is being investigated (FELICIJAN; FURBER, 2004).

Arteris (2006) is a start-up company based in Paris and founded in 2003. The company's focus is on on-chip communications design and tools. Arteris is developing tools to efficiently connect and manage the on-chip traffic requirements among all the various elements required in today's core-based SoC designs. The supported on-chip interfaces are OCP and AXI.

Clermidy et al. (2005) presented the FAUST NoC architecture, supported by ST Microelectronics, which is based on a mesh topology. FAUST supports both best-effort and QoS traffic. A NoC-based prototype for telecom applications is demonstrated. This architecture contains 23 IP connected to a 20 nodes network for a total complexity of 8 Mgates ($0.13\mu$ technology).

Lee et al. (2005) from Korea Advanced Institute of Science and Technology (KAIST) presented three test chips using $0.38\mu$, $0.18\mu$, and $0.18\mu$ technology and die sizes of $6.0mm \times 10.8mm$, $5mm \times 5mm$, and $5mm \times 5mm$, respectively.

# 3 PRIOR WORK

The goal of this chapter is to review the state-of-the-art in the SoC test field. Figure 3.1 presents a graphical classification of several problems in SoC test, adapted from (MARINISSEN; IYENGAR; CHAKRABARTY, 2002). This list is not intended to enumerate all the possible problems, but to help to identify the main problems that this thesis deals with. The problems are classified in four categories: (*i*) core test wrapper, (*ii*) test access mechanism (TAM), (*iii*) test scheduling, and (*iv*) test resource partition (TRP). The categories are sub-divided such that the sub-categories related to 'design' consist of actual design proposal and the sub-categories 'optimization' refer to finding design parameters such that 'objectives' are met under certain 'constraints'. The literature related to these four broad categories is surveyed along this chapter such that each section is related to one category. For example, test wrappers are presented in Section 3.1; TAMs are presented in Section 3.2; test scheduling is presented in Section 3.3; test resource partition, which consists of different ways to implement embedded test sources and sinks, is presented in Section 3.4. The Section 3.5 presents more specific test approaches for NoCs, interconnects, and FIFOs.

Figure 3.2 illustrates a graph of influence among the surveyed literature and the problems addressed by this thesis. It shows why certain subjects were surveyed and where this knowledge was used to develop the approach proposed in this thesis. The left side of this figure shows the sections of this chapter. The right side shows the main addressed problems. These problems are: to build a general system test model, ATE interface, DfT for routers and NIs, DfT for cores, and test scheduling.

For instance, the overall system test model, presented in Chapter 4, emphasizes the compatibility with conventional TAMs, thus, it has a stronger relation with Section 3.2. The ATE interface, the DfT module used to connect the test pins to the NoC, is related to TAMs (Section 3.2) and it is related to the different types of TAMs (Section 3.4). The proposed wrapper design for cores (AMORY et al., 2006) is obsviously related to the review of test wrappers (Section 3.1). The wrapper for routers (AMORY et al., 2005) was influenced by the study presented in Section 3.5.

## 3.1 Test Wrapper

This section describes the main functionality and the design of a test wrapper with focus on the IEEE Std. 1500 wrapper proposal (SILVA, 2005; MARINISSEN et al., 2002). Other wrapper designs, like the one presented in (MARINISSEN; GOEL; LOUSBERG, 2000), have the same main principles. The goal of a wrapper is to switch between different access mechanisms. The switching capability is

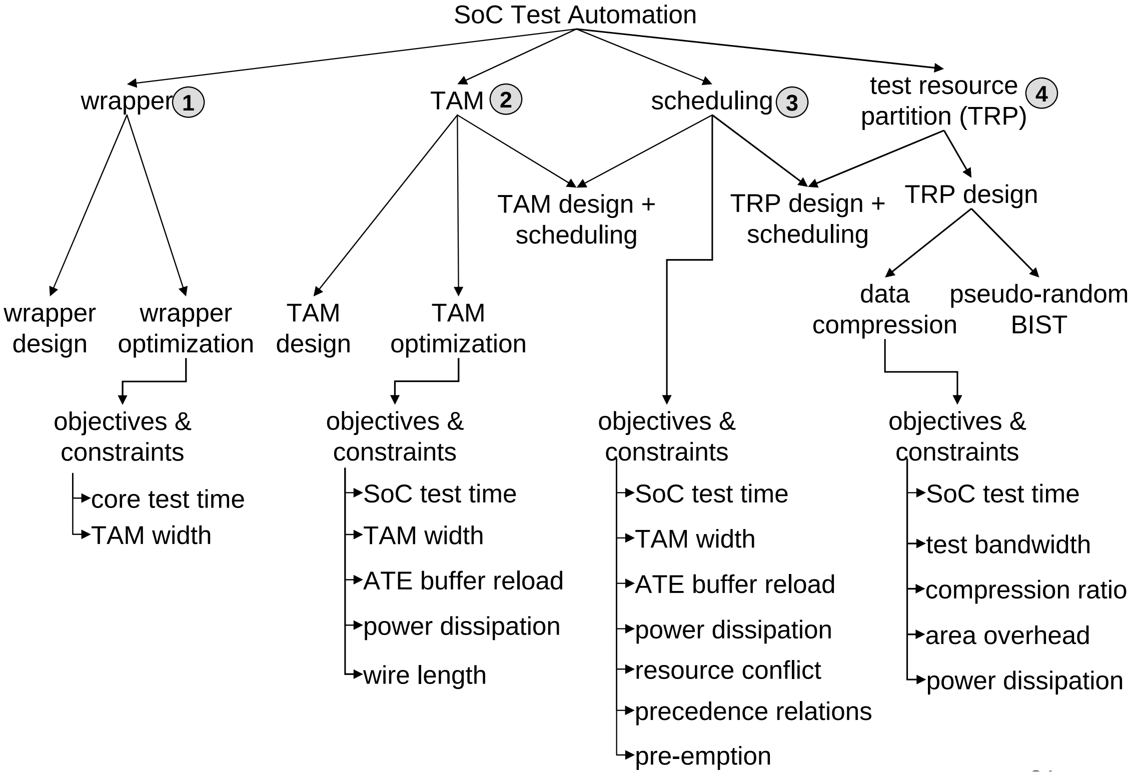


Figure 3.1: Classification of problems in SoC test (adapted from (MARINISSEN; IYENGAR; CHAKRABARTY, 2002)). The numbers identify the four main problem categories.

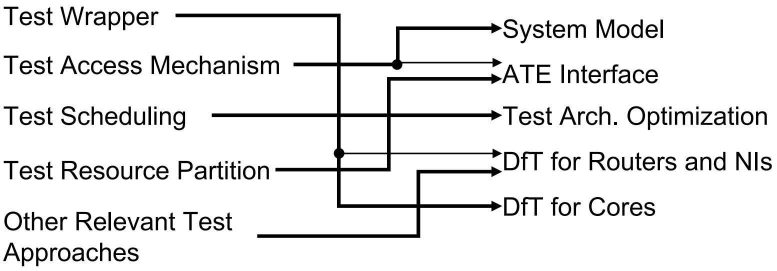


Figure 3.2: Relation among the surveyed problems and the addressed problems. The numbers represent the section number. The thicker arrow means a stronger relation.

implemented in the following modes:

- *Normal Mode* - The wrapper logic is transparent to the core, connecting the core terminals to the functional interconnect;

- *Internal Test Mode* - It is used during the actual test of the core. It configures the wrapper to access the core's primary I/O and internal scan chains via the TAM. The width of the TAM is configurable;

- *External Test Mode* - This mode is used to test logic between the wrapped cores and to test the functional interconnect. Controllability is provided from the output wrapper cells to test the interconnect logic, likewise, the observability of the interconnect responses is provided by the input cells of the wrapper;

- *Bypass Mode* - This mode is required when there are several cores connected in the same TAM. Cores already tested are bypassed to reduce the test path length.

A wrapper design problem can be informally formulated as: Design a wrapper for a given core, such that the core test length and the number of test wires is minimized. The number of test wires is a top-level constraint that is based on the number of test pins available. This problem has been identified as a $\mathcal{NP}$-hard problem (MARINISSEN; GOEL; LOUSBERG, 2000). More about wrapper optimization is presented in Section 3.1.2. Next section describes the wrapper proposed by the IEEE Std. 1500.

### 3.1.1 IEEE Std. 1500 Compliant Test Wrapper Design

Since the core-based design involves the core provider and the core user, it is required to transfer information about the core's test from the core provider to the core user. The core provider adds the appropriate core's DfT hardware (e.g. internal scan chain for random logic) and create the test patterns, while the core user integrates the test of all the cores into a top-level system test and adds the test of non-wrapped logic. IEEE Std. 1500 standardizes (SILVA, 2005; MARINISSEN et al., 2002) the test knowledge transfer and part of the test wrapper. The motivation of this standard is to enable test reuse and the test interoperability between cores from different sources. The IEEE Std. 1500 proposes a language to express test-related information of cores and a scalable wrapper design, which is the main topic of this section.

The goal of the standard wrapper is to define a uniform and flexible hardware interface to transport test patterns to a given core. The standard wrapper is required to enable an easy integration of cores. On the other hand, it must be flexible to allow the core user to explore trade-off like test quality, test length, silicon area, and performance impact. The standard wrapper must have the following features: (*i*) support multiple operational modes: normal, internal test, external test, and bypass; (*ii*) connect any number of core terminals to any TAM width. Figure 3.3 presents the structure of the wrapper:

- Functional data terminals represent the core's I/O terminals which are defined by the core provider;

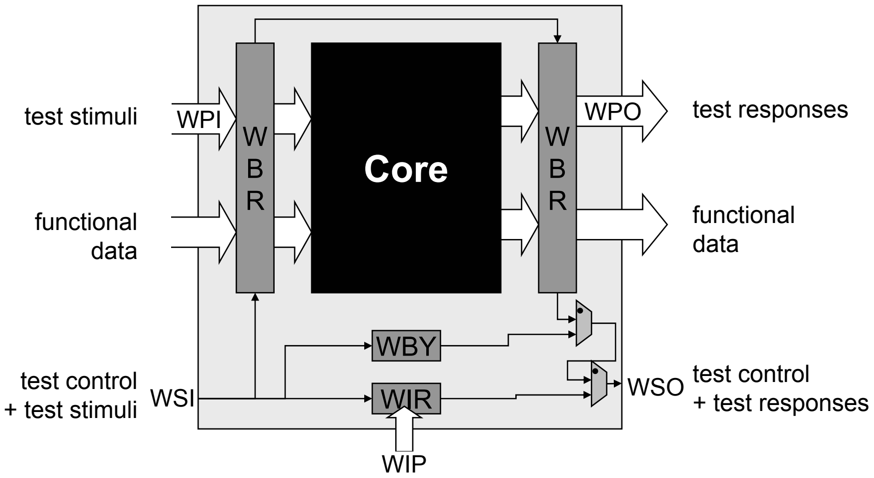


Figure 3.3: IEEE Std. 1500 wrapper architecture (from (MARINISSEN et al., 2002)).

- Wrapper Interface Port (WIP) is a six-control signals that controls the Wrapper Instruction Register (WIR). The WIR register can be loaded from WSI (Wrapper Serial Input);

- Test can be loaded serially via WSI and unloaded from WSO;

- Test can also be loaded via Wrapper Parallel Input Port (WPI) and unloaded from WPO. The width is user-defined;

- Wrapper Bypass Register (WBY) is used to bypass serial data from WSI to WSO to enable a shortened test access path. The wrapper can also have a parallel bypass register to bypass WPI to WPO;

- Wrapper Boundary Registers (WBR) consist of wrapper boundary cells that provide the test access to the core terminals. There is one cell per terminal and its basic functionality is to be transparent in functional mode and to provide controllability from the WPI and WSI ports, and observability from the WSO and WPO ports.

Figure 3.4 presents how the cores are connected at the top-level. The wrapped cores are connected to a serial TAM and, optionally, connected to a user-defined parallel TAM. The test controller may contain user-defined JTAG compatible instructions.

#### *3.1.1.1 Wrapper Example*

Figure 3.5 illustrates an example of the IEEE Std. 1500 compliant test wrapper. The core under test has two internal scan chains of six and eight bits, five primary inputs, and three primary outputs. The implemented wrapper has a serial TAM, and a parallel TAM of three bits. The wrapper cell implementation is presented in Figure 3.5(b) and (c). This implementation comprises six operational modes. Figure 3.6 presents the path activated in each of these modes and Table 3.1 presents the multiplexer configuration.

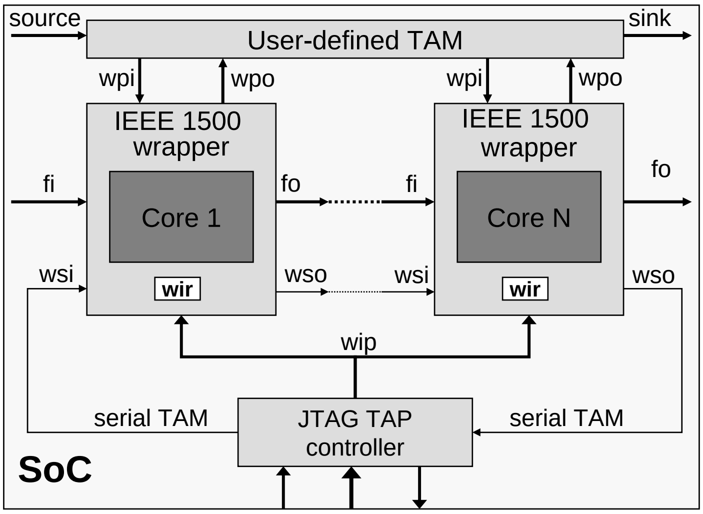
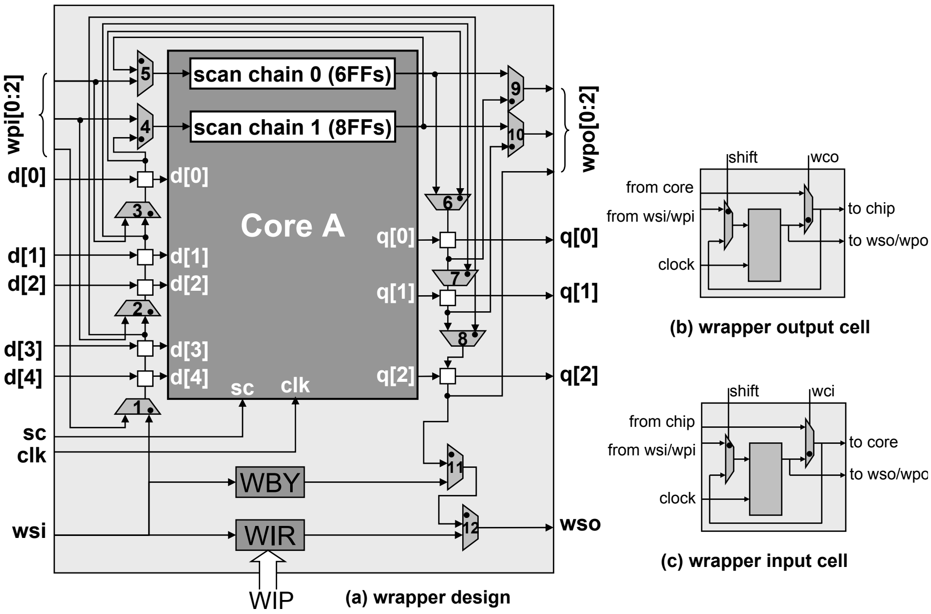


Figure 3.4: IEEE Std. 1500 wrappers at the top-level (from (MARINISSEN et al., 2002)).

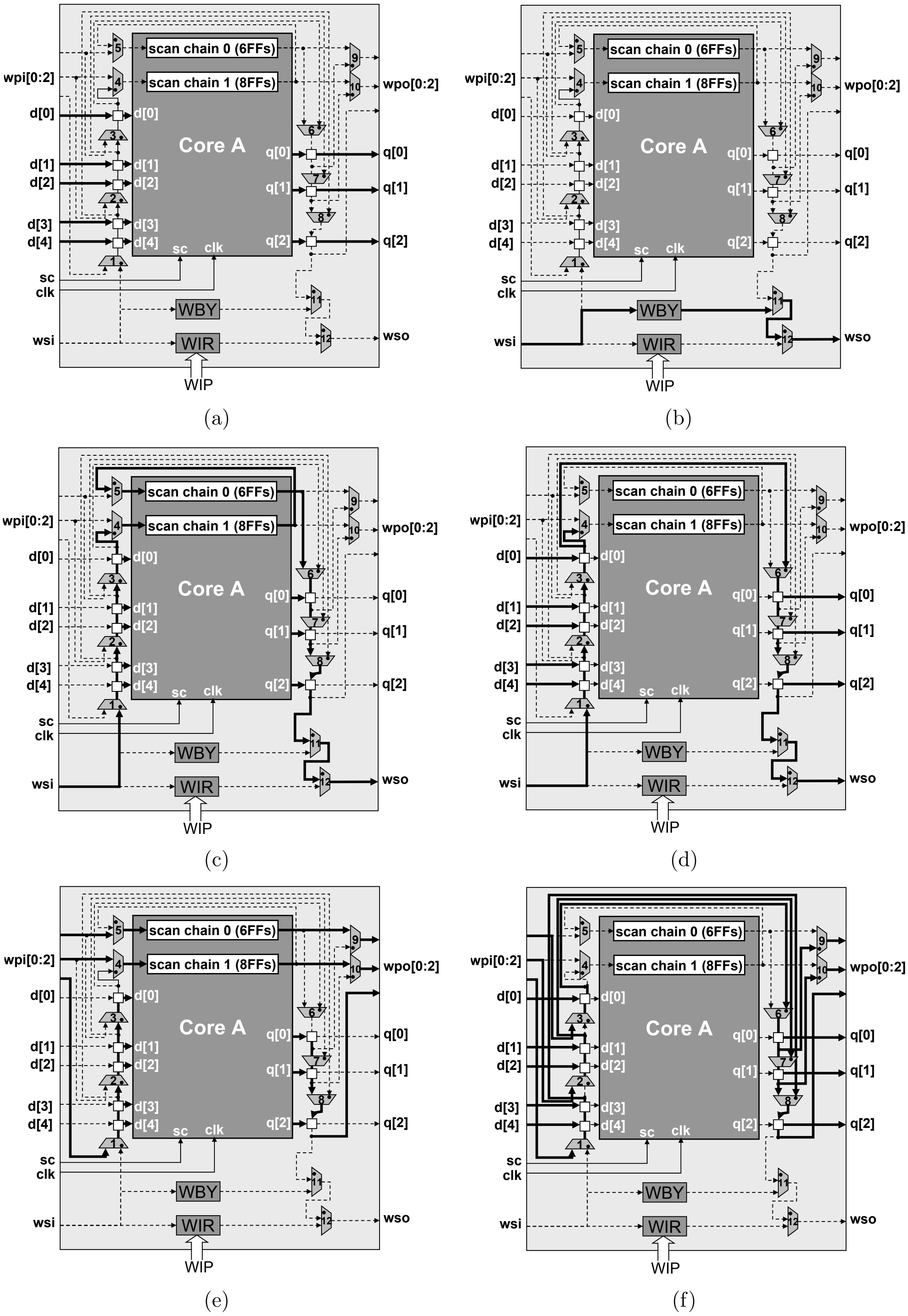


(a) wrapper design

(b) wrapper output cell

(c) wrapper input cell

Figure 3.5: Example of a IEEE Std. 1500 compliant wrapper (from (MARINISSEN et al., 2002)).



(a)

(b)

(c)

(d)

(e)

(f)

Figure 3.6: IEEE Std. 1500 wrapper's test mode. (a) normal mode; (b) serial bypass mode; (c) serial internal test mode; (d) serial external test mode; (e) parallel internal test mode; (f) parallel external test mode (from (MARINISSEN et al., 2002)).

Table 3.1: Multiplexer configuration for each test mode (MARINISSEN et al., 2002).

| mode | wci | wco | $m_1$ | $m_2$ | $m_3$ | $m_4$ | $m_5$ | $m_6$ | $m_7$ | $m_8$ | $m_9$ | $m_{10}$ | $m_{11}$ | $m_{12}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Normal | 0 | 0 | x | x | x | x | x | x | x | x | x | x | x | x |
| Serial Bypass | x | x | x | x | x | x | x | x | x | x | x | x | 1 | 0 |
| Serial InTest | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | x | x | 0 | 0 |
| Serial ExTest | 0 | 1 | 1 | 1 | 1 | x | x | 1 | 0 | 0 | x | x | 0 | 0 |
| Parallel InTest | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | x | x |
| Parallel ExTest | 0 | 1 | 0 | 0 | 0 | x | x | 1 | 1 | 1 | 1 | 1 | x | x |

### 3.1.2 Test Wrapper Optimization

#### *3.1.2.1 Wrapper for Hard-cores*

Hard cores require a wrapper since the internal DfT cannot be modified and a third party provides the test patterns. A test planning tool decides how many test wires each hard core requires and the wrapper is designed based on this number of test wires. In order to reuse the test patterns, a test protocol expansion tool translates the core-level test pattern file to the top-level tests.

The next paragraph, taken from Marinissen et al. (2000), presents the problem statement for the wrapper design of hard cores.

*Given* the number of test patterns $p$, the number of functional input terminals $i$, the number of functional output terminals $o$, a set of scan chains $S$, where each scan chain $s \in S$ has length in number of flip-flops $l(s)$. Furthermore, a number $m$ that represents the maximum number of TAM wires that can be used is given. *Determine* the wrapper design *such that* the core test length (in clock cycles) is minimized and $m$ is not exceeded.

The core test length is calculated based on the Equation 3.1

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

where $s_i$ and $s_o$ represent the scan-in and scan-out lengths, respectively, and $p$ represents the number of test patterns. Since $p$ is given, the goal of a optimization algorithm is to minimize both $s_i$ and $s_o$ to result in the minimal core test length. Figure 3.7 illustrates a wrapper test wire and its scan-in and scan-out elements. Distributing the IO terminals over $m$ TAM wires can be solved optimally in linear compute time. However, distributing the scan chains over multiple TAM wires such that the core test length is minimized is a $\mathcal{NP}$-hard problem (MARINISSEN; GOEL; LOUSBERG, 2000). Therefore, algorithms focus on the optimization of the scan chain distribution.

The most popular algorithms are Largest Processing Time (LPT) and Combine by Marinissen et al. (2000), and Best-Fit Decreasing (BFD) by Iyengar et al. (2002a). The LPT algorithm, for example, initially sorts the internal scan chains in decreasing order of length and then assigns the scan chains to the test wire with shortest length. The goal is to minimize the maximal test wire length. Figure 3.8 demonstrates an

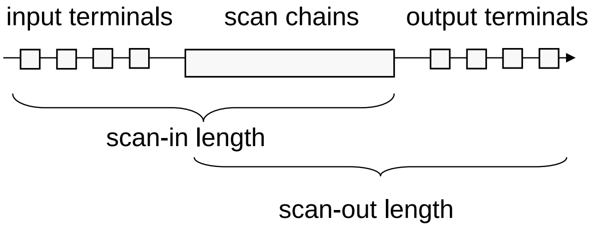
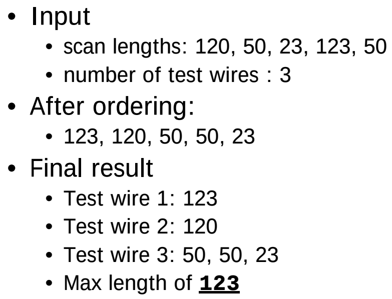


Figure 3.7: Scan-in and scan-out lengths of wrapper elements.



- Input
  - scan lengths: 120, 50, 23, 123, 50
  - number of test wires : 3
- After ordering:
  - 123, 120, 50, 50, 23
- Final result
  - Test wire 1: 123
  - Test wire 2: 120
  - Test wire 3: 50, 50, 23
  - Max length of **123**

Figure 3.8: LPT algorithm used to optimize test wrappers.

example. Suppose the core has five scan chains of lengths 120, 50, 23, 123, 50, and the test wrapper should use three test wires. In the beginning, the test wire 1 has length zero, so it receives the first scan chain with length 123. The next iteration the test wire 2 is the shortest one, then it receives the scan chain with length 120. The third test wire receives the three remaining scan chains with lengths 50, 50, and 23, which sums up 123. As a result, we get a maximal length of 123.

#### *3.1.2.2 Wrapper for Soft-cores*

The decision whether a soft core uses or not a test wrapper depends on the size of the core and if it will be reused in other designs. If it is not the case, the core can be merged with glue logic or interconnect logic, and be tested as part of the interconnect. The next paragraph presents the problem statement for the wrapper design of soft cores.

*Given* all parameters as specified in the previous problem but instead of a set of scan chains $S$ and the length $l(S_i)$ for each scan chain $S_i$, the total number of scan flip-flops $f$ is given. *Determine* the wrapper design *such that* the core test length (in clock cycles) is minimized and $m$ is not exceeded.

The scan partition problem for soft-cores is trivial. The expression $\lceil \frac{f}{m} \rceil$ finds the shortest scan chain configuration. Thus, no optimization procedure is needed to optmize wrappers for soft-cores.

## 3.2 Test Access Mechanism

The classification of TAM design alternatives illustrated in Figure 3.9 is inspired on Xu and Nicolici (2005). The first two approaches are not scalable enough for complex SoCs, thus, they are briefly presented in the next paragraph. The functional access and dedicated bus-based access approaches are presented in the following sections.

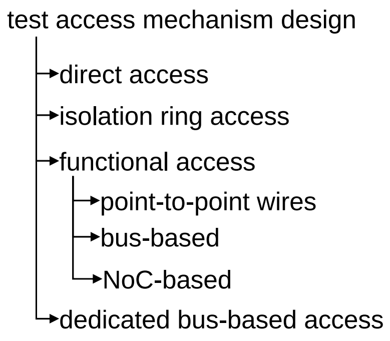


Figure 3.9: Classification of TAM designs.

The *direct access* approach multiplexes core terminals to the chip pins so that the test patterns can be applied and observed directly. However, as the number of embedded cores increases, the number of core terminals may exceed the chip pins. So, the direct access approach does not scale with the number of cores. In addition, this approach introduces a large routing overhead. The *isolation ring access* consists of isolating each core using boundary scan and serially controlling the core terminals. This approach has lower routing overhead than the direct access, but it has longer test application time due to the serial access.

### 3.2.1 Functional Access or Reuse of Functional Interconnect as TAM

There are papers proposing the reuse of different functional interconnects for test, such as, directly connected wires, buses, and NoCs. The difference between these interconnects regarding the reuse for test data transportation *motivates the study of NoC reuse rather than other interconnects.* As presented in the introduction, NoCs have the *required feature for future complex chips.* Moreover, *NoCs support parallelism*, important to reduce test length.

#### *3.2.1.1 Reusing Point-to-Point Wires*

Cota et al. (2002) present a tool that reuses *point-to-point wires.* Transparent mode is implemented in the wrapper, and mismatches between the number of functional wires and test wires are solved implementing parallel-to-serial and serial-to-parallel conversions. *The main drawbacks are the scalability in terms of number of supported cores in the chip and the wire usage of the functional interconnect, important features for future chips.* Moreover, test data is usually not pipelined, increasing the test application time.

#### *3.2.1.2 Reusing Buses*

There are several papers about *reusing buses* for test. Huang et al. (2001a) reuse a PCI bus with VCI interface. The wrapper has internal memory-mapped registers. Thus, an embedded processor can write and read these registers easily. There are some registers responsible for storing the core mode (test or normal), and registers used to store primary inputs, primary outputs, scan inputs and scan outputs. One drawback is that the approach requires a special test packet. It means that both ends of a transaction should know this special packet format, which is negative for abstraction. Their wrapper implementation requires buffers for core primary input and outputs, and buffers for scan-in and scan out to temporally store test patterns and test responses. This implementation choice increases the wrapper area proportionally to the number of I/Os and scan chains of a core (area overhead for the wrapper is not presented in the paper). Another drawback is that the buffer for primary inputs is also used by the primary outputs, which means that the wrapper does not support test pipelining.

Hwang and Abraham (2001) present a reuse model based on a Wishbone bus. They develop a test wrapper which they claim that implements the same protocol as the bus, but no details are given about the implementation, and they do not specify if it is possible to extend the wrapper to other buses and protocols. They present area and test time results, but it is compared to boundary scan, known to require a large test time due to the serialization and large area to implement its wrapper cells and control logic.

Harrod (1999) and Burdass et al. (2000) present an industrial approach for AMBA bus. The system has a module that, when it is in test mode, it acts as the bus master and it reconfigures the external bus interface to provide a high-speed 32-bit parallel interface. This approach is best suited to functional test. However, Feige et al. (1998) extend the approach to support scan-based test. An ATPG is run at the core level and a test protocol expansion tool translates the generated test patterns to the system level format, i. e. to the AMBA interface. The wrapper cells are also modified to support scan-based test. It means that both the test protocol expansions tool and the scan insertion tool are modified to support AMBA test protocol, which means that the approach has a poor backward compatibility in terms of test tools. In addition, it is a specific solution for AMBA-based systems. The authors do not claim a general bus reuse approach.

*All the above mentioned approaches suffer the common drawbacks of all reuse approaches based on buses; they require clock gating to halt the test when there is no data, the bus does not support test pipelining, and the cores must be tested sequentially. Moreover, like in point-to-point wires, buses are not scalable enough for future complex chips, and they may not provide sufficient bandwidth for the functional application. For these reasons, the reuse of NoCs for test has been studied recently.*

#### *3.2.1.3 Reusing Networks-on-Chip*

Cota et al. (2004) propose preemptive test scheduling, where the test of a core can be interrupted if there is no free path between the source to CUT or CUT to sink. A test packet can also take different paths depending on their availability, but the shortest available is selected. The drawbacks are that preemptive test may reduce the test pipelining, clock gating is required to halt the test when there is no data, and the simple wrapper employed incurs in a significant waste of test throughput, increasing the test time. Liu et al. (LIU et al., 2005) proposed non-preemptive testing. A single path from source to CUT and from CUT to sink is established in the beginning of the test and the test packets are sent one after another on this dedicated path. Although this approach preserves test pipelining, it does not guarantee that there will be a new test data in each clock cycle. For example, there are some clock cycles that are used to pack/unpack data, and also some clock cycles to execute the functional protocol. The exact timing depends on the packet format, that may be different in each design; then, the clock gating or holdable scan cells must be used.

Despite all the advantages of NoCs for both the functional and the test domains, *a NoC is more complex than a dedicated TAM.* A common aspect to all the previous approaches is the large amount of NoC implementation details required for the test model. All kinds of functional implementation details (e.g. used arbitration algorithm), temporal details (e.g. time to route a packet), and organizational details (e.g. network topology) are required, in addition to a cycle-accurate scheduling used to determine the available paths between the test source, CUT and test sink. This amount of implementation details requires major efforts to adapt the test model to different NoCs, reducing the generality of the approach. In addition, there is no detailed description of the *requirements* for test reuse, such as wrapper design and support to functional protocol. The *case study is usually a specific and simplified NoC instance* that does not represent most current NoC designs.

### 3.2.2 Dedicated Bus-Based Access

A TAM provides a path for test data transportation between the embedded cores and the test sources and sinks. In case of dedicated TAMs, this bus is used only during test application. The main *design parameters* that can be chosen by the system integrator are the number of TAMs, the width of each TAM, and the assignment of modules to the TAMs. Changing these parameters the test designer can trade-off between several test costs like DfT area, test application time, and power consumption. The *benefits* of dedicated TAM are the support to test protocol expansion (MARINISSEN; LOUSBERG, 1999) and test pattern reuse. It is also flexible because designer can trade-off, for example, test application time and silicon area by tuning the TAM width. The main benefit of dedicated TAM compared to functional access is the guaranteed test access, since the accessibility of a module does not depend on the functional chip design. The main *costs* are the design time, which is minimized with automation, and the increased routing area.

Two approaches, called Test Rail (MARINISSEN et al., 1998) and Test Bus (VARMA; BHATIA, 1998), are described in the literature. Cores connected to the same TAM based on Test Bus can only be tested sequentially. Thus, Test Bus does not support external test mode (see Section 3.1), used to test the logic between cores and the core interconnect. On the other hand, Test Rail supports external test mode and also a bypass mode, used if multiple cores are connected to the same TAM. The cores that are not been tested are in bypass mode to reduce the test path to the CUT. This mode includes a bypass register used to reduce propagation delay when there are multiple cores serially connected. Figure 2.2, page 36, illustrates a Test Rail TAM.

## 3.3 Test Scheduling

Several techniques for SoC test scheduling have been proposed for dedicated TAMs. A recent paper by Xu and Nicolici (XU; NICOLICI, 2005) surveys these approaches. A considerable amount of research has been done in this area. For example, there are scheduling considering static (GOEL; MARINISSEN, 2003a) and configurable wrappers (KORANNE, 2002a), preemptive (IYENGAR; CHAKRABARTY, 2001) and non-preemptive test (GOEL; MARINISSEN, 2003a), co-optimization of scheduling and wrapper (GOEL; MARINISSEN, 2003a), scheduling algorithms

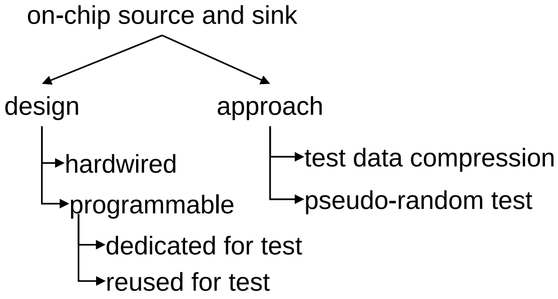


Figure 3.10: Classification of on-chip ATE approaches.

based on integer linear programming (CHAKRABARTY, 2000), based on graph theory (KORANNE, 2002b), based on packing problems (IYENGAR; CHAKRABARTY; MARINISSEN, 2002b), scheduling algorithms considering power and precedence constraints (IYENGAR; CHAKRABARTY, 2001), among several other variations. It makes sense to propose NoC-reuse approaches that can use most of this knowledge.

## 3.4 Test Resource Partition

Some types of sources and sinks are surveyed and classified in terms of their implementation as illustrated in Figure 3.10.

### 3.4.1 Off-Chip Tester

An off-chip tester can be embedded into the board of the system, where test data is loaded from, for example, a memory or a RF link. However, the most common type of off-chip test application is by means of an Automatic Test Equipment (ATE). The tester is loaded with a test program, usually generated by an ATPG tool. The test application cost is typically calculated in seconds per device for a given ATE and it determines the factory throughput, i.e. the number of chips manufactured per day. The cost of a tester depends on the number of pins and on the vector memory depth (Mbit per channel). The number of pins limits the test parallelism and the test time, while the memory depth limits the test data volume for a chip. Besides the cost of such equipment, ATE is still the most used test application approach. However, it is predicted that the increase of logic per pin and the high test frequency will require even more expensive testers, increasing the chip cost. For this reason, alternative on-chip test application methods are investigated.

### 3.4.2 Hardwired On-Chip Tester

This class consists any type of embedded tester designed and optimized specifically to test a specific set of cores (one or more cores). The tester would require redesign to test a different set of cores than the set it was designed for. This type of tester typically requires tools to generate the test logic. The most common type of embedded tester is Built-in Self-Test (BIST) related logic.

An alternative to ATE-based test application is the use of BIST, where hardware modules responsible for the test pattern generation and response analysis are embedded into the chip (BARDELL, 1987; STROUD, 2002). BIST reduces the amount of data transferred between the ATE and the chip, thus, a cheaper tester can be used. BIST enables at-speed test to increase test quality.

Hetherington et al. (HETHERINGTON et al., 1999) present results for the application of logic BIST for industrial designs. Practical issues are reported in order to generate BIST-compliant cores for at-speed testing. The paper reports fault coverage and area overhead comparison between ATPG and logic BIST. The results present that logic BIST can achieve similar test quality to ATPG with minimal area overhead and few changes to the design flow.

On the other hand, many authors say that the main problem related to pseudo-random test is the *large number of test patterns* and, in several cases, *low fault coverage*. One can argue that the issue related to the number of test patterns can be alleviated considering that, for example, the LFSR runs faster than the external tester. Related to the low fault coverage, several papers have demonstrated the advantages of using BIST schemes based on *multiple polynomials and seeds*.

Hellebrand et al. propose in (1995; 1996) a BIST scheme that supports LFSR with multiple polynomials and seeds. A set of seeds is obtained to increase the fault coverage by solving systems of linear equations. Fagot et al. (1999) propose a simulation based method to compute an efficient seed of a LFSR. The method is intended to produce a single seed that tests the hardest to detect faults of the CUT and achieves a high fault coverage. Krishna and Touba (2001; 2002) propose a loss-less test vector compression technique which combines LFSR reseeding and statistical coding to provide a high encoding efficiency.

The main problem related to reseeding approaches is the *computation time* required to find the good seeds and to find a good *trade-off between number of seeds (i.e. memory requirement) and fault coverage.*

A hardwired on-chip tester can also implement test data decompression (RAJSKI; TYSZER; ZACHARIA, 1998). This approach also enables the use of a slower external tester to provide compressed test data, which is decompressed on-chip and applied to the CUT. The test responses are compressed and sent back to the external tester.

The main challenge related to test data decompression is the *trade-off between compression rate (i.e. memory requirement) and complexity of decompression algorithms.* Very efficient compression algorithms exist but their complexity would increase the area overhead in case it is implemented in hardware, or the test time in case it is implemented in software.

### 3.4.3 Programmable On-Chip Tester

A programmable on-chip tester has the advantage that the same hardware can be used to test different cores, changing just the “test program” and test data. This type of tester can be *dedicated for test* or *reused from the functional cores.*

#### *3.4.3.1 Dedicated Programmable Tester*

Cota et al. (2001) propose an embedded test controller to test some blocks of the SoC that require at-speed testing. Such a controller, called MET, is embedded into the system with the purpose of test data manipulation and controlling. As the test controller is synthesized in the same technology of the SoC, at-speed test can be performed. The MET approach relies on the definition of test-specific instructions. Additionally, the test vectors and response are assumed to be stored in the memories that are usually available in the SoC.

Amory et al. (2004) propose a programmable LFSR and MISR with support to reseeding to increase the fault coverage. These modules were integrated to a IEEE Std. 1500 compliant wrapper in order to provide a reusable and programmable BIST approach that could be used to test a large set of cores.

Some papers (APPELLO et al., 2003; HUANG et al., 1999) propose a programmable memory BIST processors. The BIST processor can be programmed with different memory test algorithms, such as March algorithms, and be reused to test different memory cores.

Although the main motivation of programmable and dedicated on-chip testers is to reuse the same hardware for different cases, there are tools, like memory BIST and logic BIST from Mentor Graphics, that automatically insert customized BIST for a specific core. In addition, programmable tester also needs a tool to generate efficient test programs.

#### *3.4.3.2 Reused Programmable Tester*

The reuse of an embedded processor for testing other cores in a SoC has been largely studied in the literature (HUANG et al., 2001a; HWANG; ABRAHAM, 2001, 2003; AMORY; OLIVEIRA; MORAES, 2003; KRSTIC et al., 2002). The approaches proposed so far consider different aspects of the processor reuse, ranging from the reuse as test controller to the use as a built-in test pattern generator with compression features.

Huang et al. (2001a) present a bus-based architecture with a MIPS processor, a PCI bus, and VCI interfaces. Using this architecture, the authors evaluate the test time and fault coverage of some ISCAS89 benchmarks. Lai and Cheng (2001) use the same architecture presented in (HUANG et al., 2001a) to evaluate test programs generated for four ISCAS89 benchmarks, using the DLX processor. The test program length ranges from 40 to 27,000 bytes while the test length ranges from 94 to 30,430 clock cycles. The results point to important requirements in terms of test memory and test length compared with hardware-based test.

Hwang and Abraham (2001) present a bus-based architecture with an ARM processor and Wishbone interface. The authors compare the test time and area overhead between software-based test and boundary scan. In both cases, the software-based test present better results. In (HWANG; ABRAHAM, 2003), the same authors evaluate a new test pattern compression method in which test data can be decoded rapidly on embedded processors. They compare the compression results with other compression methods implemented on embedded processors. They use ISCAS89 benchmarks as CUT.

Amory et al. (2003) present a CAD tool, which helps the designer to integrate cores on a bus-based SoC and generate test programs. These test programs are loaded into the processors to test other cores. Results are presented using a subset of ISCAS benchmarks as CUT.

Marcon et al. (2004) evaluated the implementation of several memory BIST algorithms running in different processors against a hardware implementation. Test time, memory, requirements for the test code, area overhead, and power dissipation were evaluated.

In most sittuations, the reuse of embedded processor for test depends on the reuse of functional interconnect for test data transportation. All the previous mentioned approaches suppose that the chip interconnect is a bus. The problem with this assumption is, in case there are multiple processors embedded into the chip, their use for test may not represent shorter test length because the bus is the bottleneck to the data transfer. For this reason Amory et al. (2004; 2005) did the first attempt to evaluate the *reuse of multiple embedded processors for testing NoC-based SoCs.* The results show that the combined use of external test and multiple processor reuse can reduce the test length in case the tester has few test pins.

Processor reuse for test has the advantage that the “source” and “sink” are already embedded and connected to the “TAM” (i.e. the functional interconnect). Thus, their use has no additional cost. However, the main challenges for an effective reuse of embedded processors are the *trade-off between the program complexity, fault coverage, and test length.* On one hand, a very simple and fast test program with small program size can be used, but the number of required test patterns may become an issue. On the other hand, a more complex test program implementing a complex decompression algorithm requires more clock cycles to decompress a single test pattern and requires more memory to store the test program.

## 3.5 Other Relevant Test Approaches

### 3.5.1 NoC Testing

Previous papers presented the problem of NoC testing (AKTOUF, 2002; UBAR; RAIK, 2003; VERMEULEN et al., 2003). Ubar et al. (2003) and Vermeulen et al. (2003) suggested that a wide variety of standard DfT solutions can be used, from BIST for FIFOs, to functional testing of wrapped routers. However, these proposals have not been applied, to the best of our knowledge, to actual NoCs. In addition, a NoC can also be tested using standard core-based modular testing strategies (SILVA, 2005), i.e. the use of an IEEE Std. 1500-compliant test wrapper and the use of scan-based approaches to test the routers. Hence, the NoC can be considered either a flat core, i.e. a single test wrapper is inserted into the NoC interface, or a hierarchical core, i.e. additional test wrappers for each router are necessary.

Amory et al. (2005) evaluated the boundary-scan approach and the core-based modular testing in a NoC. The results presented a very high area overhead for these approaches brought from board-level testing and chip-level testing, respectively. These results motivated the *research for test approaches specific for NoCs.* Amory et al. (2005) proposed a test strategy for NoCs based on identical routers. The authors claim that the test costs (DfT area, test length, test data volume, and ATPG run) can be reduced when there are a large number of identical routers on the NoC. The paper presents a wrapper design with broadcast capability, the DfT required by the routers, and the hardware for on-chip test response evaluation. However, the approach is limited by the number of identical routers existing in the chip. In addition, output comparison may lead to a higher wiring length (WU; MACDONALD, 2003), which was not evaluated in the paper.

Grecu et al. (2005) proposed and evaluated an approach for testing the NoC routers. The approach is based on progressive test of the NoC, where the already tested NoC routers are used to transport test data to the routers under test. The approach minimizes the use of TAM to transport the test data and minimizes the test length since it explores the parallel path of the NoC and the multicast capability. The authors assume that all the routers on the NoC are identical, hence, the same test patterns are used to test them. The test length using sequential testing (i.e.

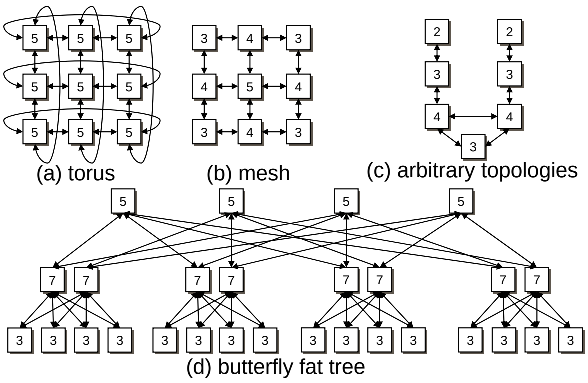


Figure 3.11: Different network topologies and their degree; torus, mesh and fat tree used in (GRECU et al., 2005), and an arbitrary topology used in (HOSSEINABADY et al., 2006). The number indicates the router degree.

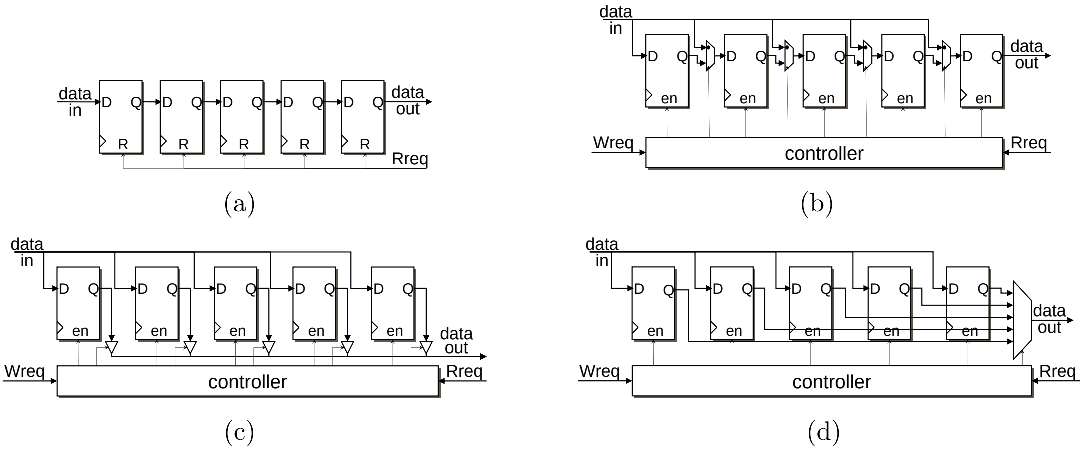
unicast) and multicast testing are compared; area overhead is not evaluated. The authors present a series of assumptions like the test response is compared locally, the existence of identical routers, the existence of multicast, and the reuse of NoC channels. Hosseinabady et al. (2006) proposed a test data broadcast approach. The authors do not focus on how the internal FIFOs and interconnect are tested. Like Grecu et al. (2005), Hosseinabady et al. (2006) did the same assumptions.

Let us compare a single feature of the presented previous approaches (AMORY et al., 2005; GRECU et al., 2005; HOSSEINABADY et al., 2006). All of them assume that *all routers are identical*. This assumption is not a problem at all, when used in the correct context. Torus topology MAY have identical routers. For example, all routers in a torus topology have the same degree, i.e. number of ports. However, the NoC designer has the freedom to specify, for example, that a specific router have bigger FIFOs than the other because it is the bottleneck of the application. That is why a torus topology MAY have identical routers. On the other hand, Grecu et al. (2005) evaluated mesh and fat-tree topologies; Hosseinabady et al. (2006) evaluated arbitrary topologies. One can not assume that all routers are identical because, as counter-example, routers have different degree in mesh, fat-tree, and in arbitrary topologies (See Figure 3.11 where the number in each router represents the router degree considering that each router is also connected to a core). Routers with different degree means that routers have different number of primary inputs and outputs, thus, different test patterns, thus, broadcast to send test stimuli is limited, thus, output comparison can not be used to evaluate all the routers. In conclusion, equal number of ports is a necessary condition for identical routers, but it is not the only condition. For example, buffer sizes and logic should necessarily be equal too.

The assumption of identical routers is just one example of simplification that can reduce the scope of the proposed test approach. It brings us the problem of finding a more general test approach, which is the main focus of this thesis.

### 3.5.2 FIFO Testing

The area for FIFO buffers corresponds to a considerable amount of the NoC area (RIJPKEMA et al., 2003; MORAES et al., 2004; SAASTAMOINEN; ALHO; NURMI, 2003). Thus, these buffers are designed as small as possible, just sufficient



(a) (b) (c) (d)

Figure 3.12: FIFO designs. (a) Conventional shift register; (b) push-in shift-out register; (c) push-in bus-out register; (d) push-in mux-out register; (from (BENINI; DE MICHELI, 2006)).

to support the functional communication requirements, in order to reduce the size of the NoC (SAASTAMOINEN; ALHO; NURMI, 2003). Other approaches use full-custom FIFOs to reduce the area (RIJPKEMA et al., 2003). Considering a $0.38\mu m$ technology library, the area of a SRAM cell is $84\mu m^2$ while the area of a DFF is $984\mu m^2$ (BENINI; DE MICHELI, 2006). For performance reasons, the NoC designer can use FIFOs on the NoC in different positions (in the inputs, outputs, or even centered). Moreover, different FIFO designs can be used in NoCs (see Figure3.12); each one requires different fault models and BIST circuitry. One should not forget that the NoC is composed of dozens of small FIFOs distributed over the chip. These characteristics complicate the development of a general test approach for the NoC FIFOs.

The most common FIFO designs are shifting-type FIFOs and the RAM-type FIFOs (GOOR; SCHANSTRA; ZORIAN, 1995). The former is based on a shift register that shifts data from the write port to the last unused location. The latter has a read and write address register to access data. RAM-type FIFOs can also be implemented as single-port or dual-port FIFO.

The papers about FIFO testing can be classified as: fault model proposal, BIST proposal, or scan based. Van de Goor et al. (1995) proposed a fault model for shifting-type FIFOs. Aitken (2004) proposed a modular wrapper for at-speed BIST and repair of small memories. The approach is specially interesting for a large number of distributed register files and FIFOs. The DfT overhead presented for a 64x64 FIFO is 28%, while for smaller FIFOs the area overhead increases even more, which can be considered high for NoCs. The area overhead can be reduced if the logic related to memory repair is removed.

The main drawback of BIST circuitry to test small and distributed memories is the large area overhead. The memories may be distributed throughout the chip to allow sharing the BIST circuitry. An alternative approach, presented in (REARICK, 1999; AMORY et al., 2005), is defining a general DfT for FIFOs. Both papers propose the insertion of scan chains around the memory matrix, i.e. scan chain in the data input, data output, and address registers. Thus, any test algorithm, like the one proposed in (GOOR; SCHANSTRA; ZORIAN, 1995) to test the memory

Table 3.2: Properties of interconnect test sequences ( from (JUTMAN, 2004)).

| | count. seq. | true/compl. seq. | interl. true/compl. | walking seq. |
|---|---|---|---|---|
| # of vectors | $\lceil \log_2(N) \rceil$ | $2 \times \lceil \log_2(N) \rceil$ | $2 \times \lceil \log_2(N) \rceil$ | $N$ |
| defects | shorts | shorts, opens, (delays) | shorts, opens, delays | shorts, opens, (delays) |
| diag. | bad | good | good | best |

matrix, can be employed via scan chains to test the FIFOs. Scan based test of FIFOs increases the test length, but the area overhead is significantly reduced.

### 3.5.3 Interconnect Testing

Testing and diagnosis of NoC channels is also important. It is highly recommended to use delay fault model to test the global interconnects which may have long wires. *Delay fault testing* requires a sequential application of two vectors such that the path through the logic is set-up by the first vector while the second vector produces the transition for the delay fault detection.

We highlight two approaches for that: *at-speed interconnect BIST* and *scan-based delay test*. Jutman (2004) described a new test sequence, called Interleaved True/Complement code, and BIST circuit for interconnect. Test length and detected faults of the proposed test sequence were compared to sequences like Counting Sequence, Walking One, and True/Complement (Table 3.2). The proposed approach detects both static and dynamic faults on interconnects and it supports at-speed diagnosis without aliasing. The proposed parallel test pattern generator and on-chip at-speed response analyzer are very efficient in area. Jutman et al. (2005) applied the approach proposed by (JUTMAN, 2004) for testing the NoC interconnect. However, the test of the flow control wires of a channel was not addressed. In addition, a complete NoC may have several channels and the total overhead for the interconnect BIST may become relevant. The second problem is that typically the channels does not work at-speed, but according a certain protocol. For example, handshake typically takes two clock cycles to perform a transaction. Running the test faster than the functional speed may over-test the wires. Moreover, power consumption may become an issue while testing several channels in parallel and at-speed.

The second usual approach is a *scan-based delay testing*, called *enhanced-scan test* (BUSHNELL; AGRAWAL, 2000), which requires scan cells with an extra latch called hold latch. This latch temporally stores the second vector used for delay fault detection. A recent approach called *first level hold* proposes a delay fault testing technique, which allows enhanced scan-like test application, but with lower area overhead (without the hold latch) (BHUNIA et al., 2005). The approach holds the state of the combinational circuit by gating the VDD and GND of the first level logic gates. It does not require extra area and signals, it does not change the test generation/application process, it reduces the delay overhead and power compared to enhanced-scan approach.

The test length of scan-based approaches requires shifting test patterns, then they have higher test length than interconnect BIST approaches, however, scan-

Table 3.3: Comparison between the TAM approaches.

| | application-independent | support parallelism | simple design | trade-off test costs | future chips |
|---|---|---|---|---|---|
| point-to-point | no | yes | no | ? | no |
| bus | yes | no | yes | no | no |
| NoC | no | yes | no | ? | yes |
| dedicated TAM | yes | yes | yes | yes | ? |

based approaches have lower area cost.

## 3.6 Discussion

Since the main goal of this thesis is to reuse the NoC in a cost-effective manner, we concentrate the discussion on different TAM approaches. Table 3.3 compares five TAM features among the reviewed TAM approaches.

*Application-dependency* informs how likely the TAM design changes according to the application requirements (performance and cost). The support to *parallelism* tells if the TAM supports independent communication flows in parallel and supports high bandwidth. *Simple design* measures whether it is easy to integrate the test approach into the overall SoC design flow. The ability to tune the TAM design according to test requirements is indicated in *trade-off test costs*. *Future chip* informs how likely the TAM can be used in complex chips using future technologies.

A comparison between the reuse of these three types of functional interconnects reveals that NoCs have two striking advantages compared to buses and point-to-point: NoCs are more likely to be *supported by future complex chips* (BJERREGAARD; MAHADEVAN, 2006; JANTSCH; TENHUNEN, 2003; BENINI; DE MICHELI, 2002; GUERRIER; GREINER, 2000; ZEFERINO; SUSIN, 2003) and NoCs *support parallelism* (see hypothesis 1, page 69).

The comparison between NoC and dedicated TAMs tells that both of them support parallelism to reduce the test length. On one hand, dedicated TAMs are based on a very simple design, they are application-independent, and they allow trade-off test costs (XU; NICOLICI, 2005). However, it is unclear if dedicated TAMs can be efficient in future chips because of the routing overhead and long global wires (see hypothesis 2, page 69).

Concerning NoCs, as far as we know, there is no work demonstrating that it is possible to trade-off test costs in NoC architectures and none of the existing approaches was able to abstract the NoC complex and application-dependent design. We believe that if these three issues can be overcome, then NoC reuse might be a competitive approach.

# 4 PROPOSED TEST MODEL

This chapter introduces the proposed test model. It presents hypothesis, required properties, test requirements, the sources of jitter in NoCs, the proposed test model, an overall problem statement, and the proposed test flow.

## 4.1 Fundamental Hypotheses

The proposed test approach presented in this thesis relies on two fundamental hypotheses:

*Hypothesis 1: NoC will be the main communication architectures for future and complex chip designs.* There are several papers in the literature demonstrating advantage of NoCs compared to other communication architectures. In addition, as demonstrated in Section 2.2.2, there are several industrial approaches coming up and, recently, a test chip based on NoC has been announced (CLERMIDY; VARREAU; LATTARD, 2005).

*Hypothesis 2: Can dedicated TAM be the main test access mechanism for these future and complex chip designs ? The answer is probably not.* The reason for this is that traditional global on-chip interconnects, like buses (a dedicated TAM is a kind of bus), are becoming the bottleneck in terms of bandwidth, signal integrity, and power dissipation of complex chip designs (ITRS, 2007).

In addition, one can observe that most test planning algorithms proposed in the literature minimize the chip test length, but minimization of TAM design costs (e.g. wiring length) has received little attention; they are based on layout constraints to minimize the TAM wiring length. For example, according to Goel and Marinissen (GOEL; MARINISSEN, 2003b) the wiring length can be reduced about 83% with a test length penalty of 4%. However, these approaches assume that the layout position of every core is known a priori. This assumption complicates the design flow since, in the early stages of design, the layout information is not available. To get the information it would be required to build the layout of the full chip, then return to the initial design steps to insert the TAM constrains and re-implement the layout with optimized TAMs.

In conclusion, approaches for wire length minimization of TAMs are required for complex chips, nevertheless the approach based on core layout position complicates the design flow. New approaches for TAMs are required.

## 4.2 Properties of Proposed Test Model

Building a general NoC reuse approach for BE NoCs is the main concern of this thesis. We have identified in the Prior Work that one of the main problems with most reuse approaches is that they are *application-dependent* and not *general* since the interconnect design is typically either application-dependent or complex.

Our second main concern is compatibility with the conventional test model based on dedicated TAM. We enforce this compatibility because, based on the conclusion of the Prior Work section, the SoC test approach based on dedicated TAM seems to be the most cost-effective approach so far. In addition, we build the proposed test model on top of existing and well-established research and standards. Thus, we can use the same optimization procedures, DfT logic, and tools for the proposed approach. In fact, few adaptations for the DfT logic and the algorithms are required, but we reuse the core optimization algorithms and DfT design strategies. These adaptations are one of the main contributions of this thesis.

The remaining concerns and required properties are:

(*i*) The *trade-off of test cost parameters* to tune the approach under several constraints. The proposed approach optimizes DfT area and test application time. Other cost functions like power consumption can be included in future work;

(*ii*) The DfT modules should be connected as plug-and-play, without modifying any other module except the DfT modules;

(*iii*) In addition, the DfT modules should have *minimal impact on the functional requirements* and the proposed method should *fit easily in a chip design flow.* Recall that, as seen in Section 4.1, the dedicated TAM approach with layout constraint complicates the design flow;

(*iv*) The *support to multiple TAMs* to allow testing multiple cores in parallel reducing the test application time.

The six classes presented below summarize the required properties of the proposed approach:

1. general and application-independent test approach;
2. Compatibility with conventional test approach;
3. Enable design-space exploration and trade-off test cost parameters;
4. Fit easily in a chip design flow;
5. Minimal influence on the functional requirements;
6. Support to multiple TAMs.

## 4.3 Test Traffic Requirements

*Test traffic requires uncorrupted, lossless, in-order data delivery, zero-jitter, and scalable and constant bandwidth.* We want to keep these requirements to preserve the natural test streaming expected by the ATEs and the CUTs.

The proposed approach requires that the NoC provides the previous mentioned requirements. The first two requirements are usually implemented in NoCs. The third one is easily accomplished by sending test data over a single path. Jitter is caused because NoCs have several shared resources (channels and routers). Shared resources require arbitration schemes, which cause jitter. Scalable and constant bandwidth is easily solved if the zero-jitter requirement is fulfilled; it only requires to send/receive data at regular time intervals.

In conclusion, zero-jitter requirement is the most critical to be met. The next section identifies the sources of jitter in NoCs.

## 4.4 Sources of Jitter in NoCs

### 4.4.1 Shared Channels

Channels are shared in a network to save area for wiring. However, when two packets compete for the same channel, the arbitration logic grants access to one of them. The other stream is delayed, causing jitter and disrupting the test application of a core.

### 4.4.2 Shared Routers

The routing time of a router, i.e. the time it takes to find the output port, also causes jitter. There are two types of router schemes that are relevant for test: *routers with distributed and centralized routing* (DUATO; YALAMANCHILI; NI, 2003).

In the first case, the routing logic is typically located at the input ports, thus, it usually requires more silicon area to replicate the routing logic to every input port. However, it can route several independent packets (packets that do not require the same output channel) in parallel without any loss of performance. Therefore, the routing delay is deterministic and known a priori.

Centralized routing schemes save in silicon area but cause an indeterminate time to route a given packet. Usually, even if two packet request different output channels, the packet that arrives first to the router is routed first, while the other routing requests wait. Therefore, the routing delay varies according to the router utilization.

Since it is not up to the test engineer to decide whether centralized or distributed routers are used in a design, the test strategy must support both types of routers.

### 4.4.3 Load Fluctuation

Figure 4.1 illustrates the test data flow from the input test pins to the input of the CUT. The input test pins receive continuous test data from the tester (line 1) and the CUTs expect the same traffic shape (last line). However, one can realize that the traffic shape at the input of router 1 (line 2) is different from the wrapper input (line 5). This deformation is called *load fluctuation* and it is caused by the routing delay, which in this example is four clock cycles. Thus, the data is ‘condensed’ every router, and, at the end of the flow the traffic has a bursty format (line 5). This traffic shape is not interesting for test because it creates gaps (line 5) in the data delivery. Then, our solution uses a FIFO buffer to transform the previous bursty format to the periodic format (line 6). The periodic traffic shape is preferred because it is easier to transform it to the original traffic shape sent by the tester (lines 1 and 7).

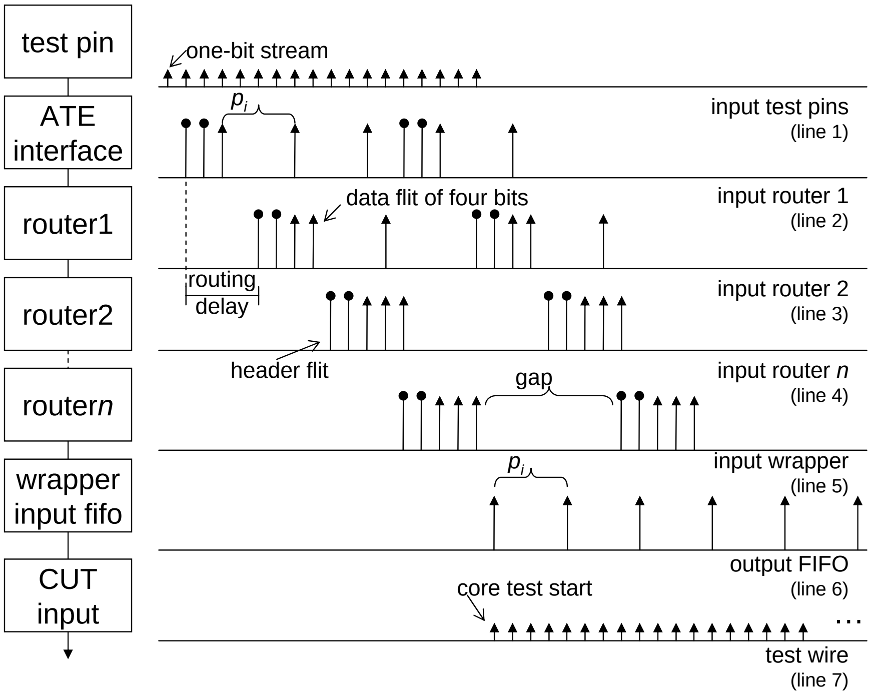


Figure 4.1: Example of load fluctuation in an input test path. $p_i$ is defined in Equation 5.1, page 86.

## 4.5 Introduction to the Proposed Model

This section briefly introduces the basic DfT modules and the NoC partition method.

### 4.5.1 ATE Interface

An *ATE interface* (AMORY et al., 2007) is the DfT module that connects the test pins to the NoC. In realistic designs, NoCs use standardized on-chip protocols, like OCP (OCP-IP, 2003), different from the test protocol used by ATEs. Thus, some protocol conversion logic is required. Recall the discussion in Section 2.2.1.2, page 45.

The main tasks of an ATE interface are to do *width conversion* between the number of test pins and the network channel width, *protocol conversion* to communicate the ATE with the NoC, and *traffic shaping*, i.e. to correct the traffic deformation caused by jitter. Section 6.2 presents more details about the ATE interface design.

### 4.5.2 Test Wrapper

Test wrappers are required to all modules of the chip along with, including cores, routers, and NIs (it may be convenient to combine a router with its NI to reduce the total number of wrappers). The basic principle of conventional test wrappers remains the same. The main difference is that the network transports the test data. It implies that the wrapper has the same three tasks of the ATE interface (width conversion, implement the on-chip protocol, and traffic shaping) plus the tasks of a conventional wrapper. Chapter 5 details the wrapper design and optimization.

The basic and common approach to design wrappers for NoC reuse is to define an input port, where test data comes in, and an output port, where the test data goes out the module (AMORY et al., 2007). The words received from the source in the input port are shifted to the remaining core terminals and scan chains. After loading the complete stimuli, the test is applied and the responses are captured in the output wrapper cells and in the scan chains to be shifted out to the output port. The output port sends the response to the sink. The wrapper must know the network address of the sink to build a correct test response packet.

The structure of wrapper for routers is the same, but it uses different protocols, which requires slightly different wrapper control logic to execute the protocol. For instance, cores use OCP on-chip protocol while routers use credit-based router protocol.

### 4.5.3 The NoC Partition Method

Basically, there are two approaches to guarantee the zero-jitter requirement: *one can use NoC with guaranteed jitter (approach 1) or DfT must be added to the NoC to provide the jitter guarantee (approach 2).* The former requires less area for DfT, but it requires more complex NoCs. Moreover, the NoC reuse can only be used in NoCs with such guaranteed jitter service, reducing the scope of the approach. The latter approach (adding DfT) naturally requires more silicon area for DfT than the former one, however, it can also be used on low cost network without any guaranteed service.

Amory et al. (2007) presented a wrapper design and optimization procedure for NoC with guaranteed jitter. The same authors have also demonstrated that, compared to the approach 1, the approach 2 requires only a small FIFO in each wrapper input (AMORY et al., 2007).

The approach proposed by this thesis is based on the second approach. In summary, the proposed DfT is built to avoid resource competition and packet collision, which are the main sources of jitter, in NoCs without guaranteed services.

On the other hand, the proposed DfT modules can be easily adapted for the first approach by removing the FIFOs. The FIFOs are still required to remove load fluctuation, even considering NoCs with guaranteed jitter. However, in most cases these FIFOs are located inside the NI (RADULESCU et al., 2005), then, they are not required in the wrapper.

The proposed solution is to sub-divide the network into different logic partitions. Suppose, that a directed graph models the entire NoC-based chip design, such that routers, NIs, and cores are nodes and the channels are edges. A *NoC partition* is a sub-graph of the system graph such that *there are ATE interfaces, which represents the test source and sink in the partition*, and *all nodes of the partition can be reached from these sources and sinks.* A module belongs to only one NoC partition and every module must be in one partition. The test inside a NoC partition is serial in order to avoid resource competition that causes jitter. The test data is transported only within a single partition and it cannot have access to the other partitions also to avoid possible resource competition.

The modules of a NoC partition are tested sequentially. However, the entire system can have multiple NoC partitions, supporting parallel testing. Figure 4.2 presents a SoC and its test partitions. This figure is a simplified view of an entire system where we can see only the routers. In fact, this figure is useful to show the partitions. When a router is part of a partition, it means that the router and all nodes attached to it, like its NI and the cores attached to the NI, are also part of

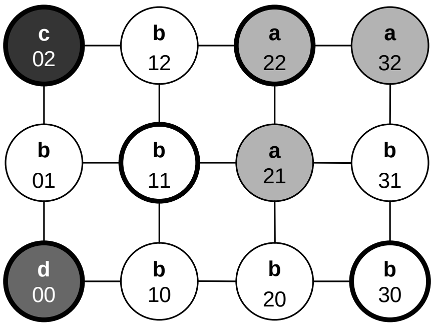


Figure 4.2: Example of partitioned NoC. The letters identify the partitions, the numbers refer to the routers, and the circles with fat lines indicated where the ATE interfaces are located.

this partition.

For instance, partitions $a, b, c$, and $d$ in Figure 4.2 are running in parallel. The ATE interfaces of partition $b$ are located in routers 11 and 30. Two ATE interfaces are required because, assuming XY [1] routing algorithm is used, there is no single position where the ATE interface would have access to all nodes of partition $b$. In this case, the ATE interface 11 tests routers 01, 11, 12, while the ATE interface 30 tests 10, 20, 30, and 31. If more than one ATE interface is required for a partition, then only one is activated at a time to avoid packet collision an also because these ATE interfaces share the same test pins.

### 4.5.4 More Examples about the NoC Partition Approach

Let us take the system illustrated in Figure 4.3(a) as an example. Figure 4.3(b) shows three possible NoC partitions.

Considering a NoC partitioning, the modules between the source, a CUT, and the sink represent a *NoC test path*. Let us take *core2* in the NoC partition 1 as an example. The NoC test path required to test this core is illustrated in Figure 4.3(c). This figure says that, assuming that its ATE interface is connected to the *ni1* and considering a given network routing algorithm, the test stimuli pass through *ni1*, *router1*, *router2*, *ni2*, and then it reaches *core2*. The responses path works in a similar manner. The same concept also applies to reach network modules like routers or NIs. For example, Figure 4.3(d) illustrates the test path to test *router9* in the NoC partition 1. However, note that in this example the stimuli path and the responses path do not necessarily consist of the same modules.

To build the test paths of a partition, the partition can be modeled as a directed graph, where vertices are routers, NIs, or cores, and the edges are determined by the physical direction of the data flow and the routing algorithm. The routing algorithm is important because the test model must not allow impossible NoC test paths since the test data is sent while part of the network is in functional mode (i.e. respecting a routing algorithm). Figure 4.3(e) represents the test paths to test the modules of partition 1 considering the XY routing algorithm. This is the reason that, due to the constraint of the XY routing algorithm, *router9* has *router2* as an incoming router and *router4* as an outgoing router. Note that the graph in Figure 4.3(e) has a relation of partial order. The partial order is required because the modules in the test path of a given CUT must have been tested first. This figure says, for example,

[1] The packet is routed in the horizontal axis first, then it is routed in the vertical axis.

that:

- the first modules to be tested are either $c_1$ or $ni_1$;
- to test $r_1$, $ni_1$ must be tested first;
- to test $c_2$, the $ni_1$, $ni_2$, $r_1$, and $r_2$ must be tested first. Figure 4.3(c) shows the stimuli path and the responses path for $c_2$ considering a XY routing algorithm;
- to test $r_9$, the $ni_1$, $r_1$, $r_2$, and $r_4$ must be tested first. Figure 4.3(d) shows the stimuli path and the responses path for $r_9$ considering a XY routing algorithm. Note that in this case the stimuli path and responses path do not involve the same set of modules.

The total number of test pins is distributed among the test partitions and the partitions can receive different number of test pins, according to the test data bandwidth required to test the modules within the partition. *The number of partitions, the modules in each partition, and the number of test pins assigned to each partition are the main problems to be solved such that the chip test length is minimized.* Figure 4.3(f) shows a valid test scheduling for this example design.

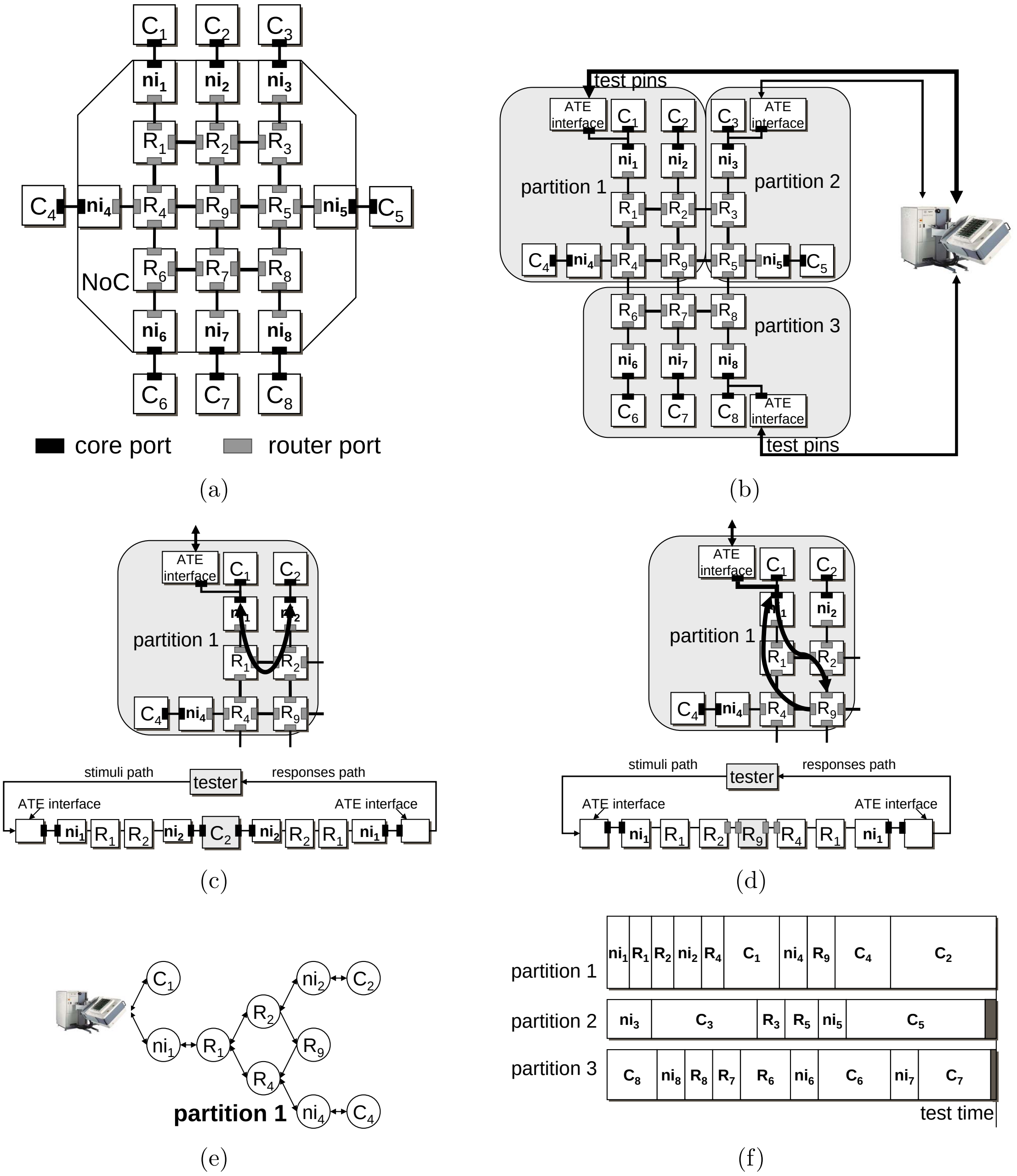
Figure 4.4(a) illustrates a typical test data flow in the proposed approach. Let us assume that the stimuli data set of a certain core consists of a set of $m$ flits and the responses data set consists of a set of $n$ flits. Then, the latency of all words of a stimuli data set ($L^s$) must be equal ($L^s_0 = \cdots = L^s_{m-1}$, thus, zero-jitter). The value of the latency does not matter for the test model (Section 4.9 explains the impact of latency for test). The latency of all words of a responses data set ($L^r$) must also be equal ($L^r_0 = \cdots = L^r_{n-1}$), but not necessarily equal to the latency of stimuli data set because the number of routers of both paths can be different. Of course, the latency to test a different core can also be different because the test path is different (different number of routers). With this timing characteristics, the NoC partition works as a pipeline (Figure 4.4(b)) driving test data to its cores, exactly like dedicated TAMs and external testers.

*The most important feature of this model is that there is only one active test source and sink per partition, which test just modules within its partition, causing no packet collision. Once there is no packet collision, complex dynamic network issues like packet scheduling, packet congestion, and cycle accurate network model are not required. Few design information is required to model the system, what makes the model simpler and more general.*

## 4.6 Reducing the Jitter Bound Using NoC Partitioning

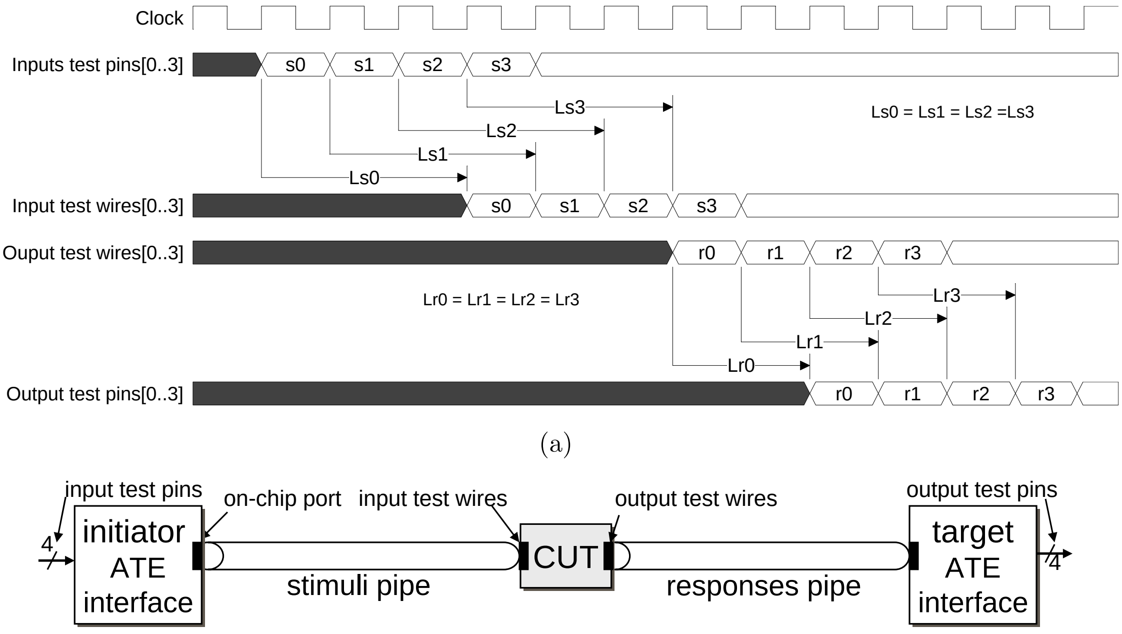
This section shoes how the NoC partition method solves the zero-jitter problem.

FIFO buffers (line 6 of Figure 4.1) are typically used to eliminate the jitter. However, the size of the buffer is proportional to the jitter bound (ZHANG, 1995). Thus, a small jitter bound implies in a lower silicon area for DfT. The rest of this section shows how the proposed approach reduces the jitter bound and minimizes the silicon area for DfT.



(a) (b) (c) (d) (e) (f)

Figure 4.3: Test model based on NoC partition. (a) an example of NoC; (b) a valid NoC partition; (c) test path for core $c_2$; (d) a test path for router $r_9$; (e) dependency graph for partition 1; (f) the test scheduling.



(a)

(b)

Figure 4.4: NoC abstracted as a pipeline and the required test data timing.

### 4.6.1 Shared Channels and Routing Logic

The proposed NoC partition solves the problem of shared channels because there is only one active test source and sink (the ATE interface) per partition causing no competition for channels and packet collision. With the proposed partition method it is possible to give guarantees of jitter even in networks without guaranteed services because the proposed approach avoids the main source of jitter, i.e. shared channels.

The NoC partition approach also helps to solve the shared router problem because no more than two flows compete for router services: the stimuli test flow and the responses test flow (Figure 4.4(b)). Thus, the number of flows competing for routers is bounded to two flows.

### 4.6.2 Load Fluctuation

While NoC partition is used to reduce the jitter bound, the jitter is actually eliminated with FIFO buffers at the end of the data flow (both stimuli and response data flows). The goal of the FIFO buffer is to receive a variable input rate (line 5 of Figure 4.1) and to create at the output a constant rate (line 6 of Figure 4.1). The FIFO works as an elastic buffer in which the data is temporarily stored and then retransmitted at a constant rate.

## 4.7 Overall Problem Statement

First, we have to define formally the system functional model described in Section 2.2.1, page 43. The system can be modeled as a weighted directed graph. A *weighted directed graph* $G$ is a pair $(V, E)$, where $V$ is a set of vertices, and $E$ is a set of edges between the vertices $E = \{(u, v)|u, v \in V\}$. In addition, there is a function $w(v)$ that represents the weight of the vertices such that $v \in V$ and $w(v) \in \mathbb{N}$. The vertices are classified as NI, router, or core. The edges represent the router ports and core ports. Finally, the weights represent the test data volume (i.e. number of bits to test a core) required to test each vertex.

In addition, a virtual TAM (vTAM) is defined as a tuple $\{d, k, w, R_{atei}, R_{part}\}$, where $d$ is the FIFO depth of the DfT modules in $R_{part}$, $k$ is the packet size used during test, $w$ is the number of test wires used to test the modules in $R_{part}$, $R_{atei}$ is the set of ATE interfaces, and $R_{part}$ is the set of routers in the partition such that $R_{atei} \subset R_{part} \subset G$. The test architecture optimization problem must determine: $(i)$ the vTAMs (the FIFO depth, the packet size, the number of test wires, the set of modules, and the set of ATE interfaces of each vTAM), $(ii)$ the wrapper design for the modules, and $(iii)$ the ATE interface design for the vTAMs.

**Problem 1** [Test Optimization of NoC-Based Chips]
Given:

- a graph $G$ which defines the SoC where for each $g \in G$:
  - there are a test input $i_g^p$ and a test output port $o_g^p$;
  - number of test patterns $p_g$;
  - number of functional input terminals $i_g$;
  - number of functional output terminals $o_g$;
  - number of functional bidirectional terminals $b_g$;
  - the number of scan chains $s_g$ for each scan chain $k$ with scan length $l_{g,k}$ (for hard cores only);
  - the number of scan flip-flops $f_g$ (for soft cores only).
- the maximal number of test wires $w_{max}$;
- the physical channel width $c$, in bits;
- the routing algorithm $r()$ used by the NoC.

Determine:

1. the sets of vTAMs;
2. the wrapper design for each module;
3. the ATE Interface design.
4. the SoC test length, in clock cycles.

Such that:

*i.* the SoC test length and the silicon area for DfT are minimized while $w_{max}$ is not exceeded and the width of each vTAM is $\leq c/2^2$.

□

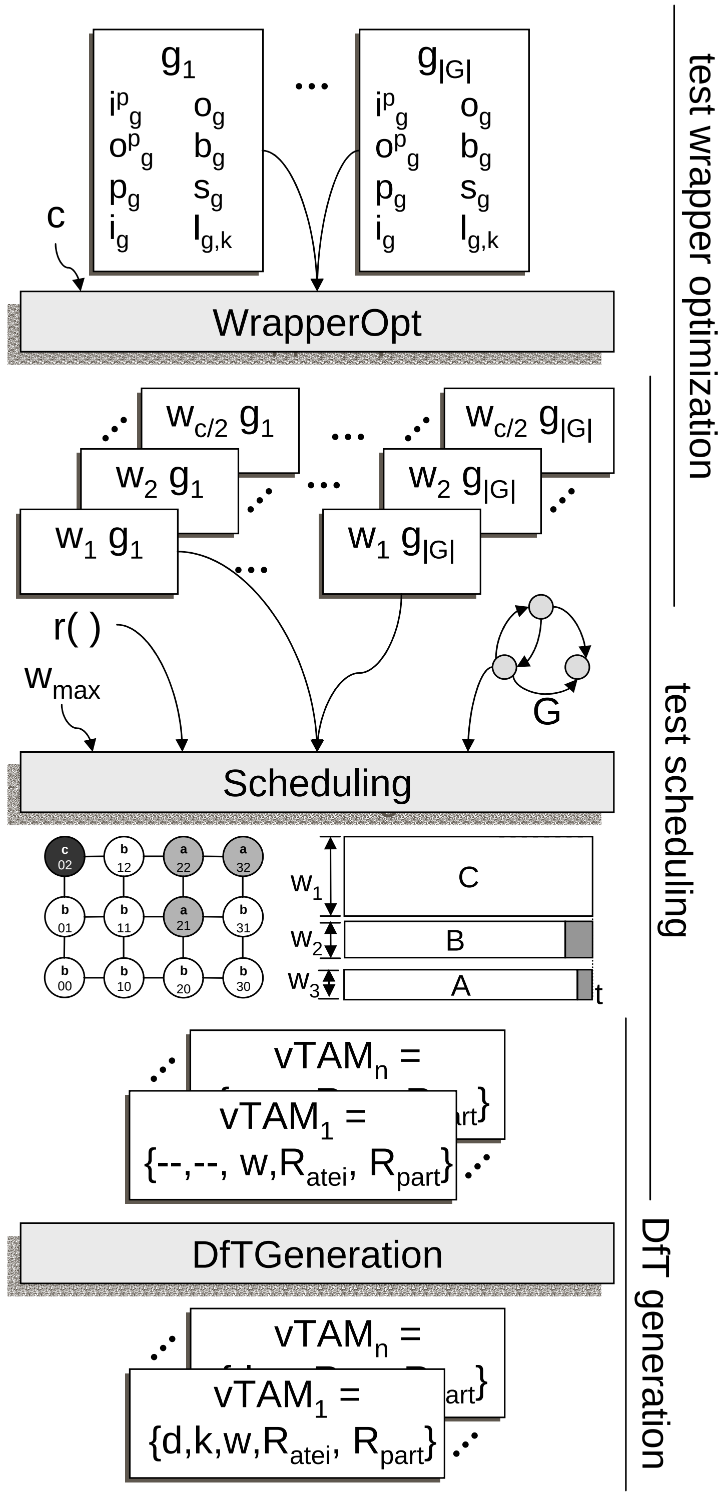


Figure 4.5: Design flow in terms of the variables used in the problem statement. $w_1 g_1$ refers to the wrapper of the module one which uses one test wire.

## 4.8 Proposed Design Flow

Figure 4.5 illustrates the proposed design in terms of the variables presented in Section 4.7. It also presents how the overall problem is divided in three subproblems: wrapper optimization, test scheduling, DfT optimization and generation.

The wrapper optimization algorithm for NoC reuse (Chapter 5 and (AMORY et al., 2007)) optimizes test wrappers for all modules considering vTAM widths from 1 to $c/2$ test wires. The wrapper optimization takes $i^p_g$, $o^p_g$, $p_g$, $i_g$, $o_g$, $b_g$, $s_g$, $l_{g,k}$, and $c$ of each module $g \in G$ to generate the $c/2$ optimized wrappers for each module. For instance, $w_1 g_1$ means the wrapper with one test wire for the module $g_1$, $w_1 g_{c/2}$ means the wrapper with 1 test wires for the module $g_{c/2}$, $w_2 g_{|G|}$ means the wrapper with two test wires for the module $g_{|G|}$.

The optimized wrappers for all modules, $r()$, $w_{max}$, and $G$ are used by the scheduling tool (Chapter 8) to determine the vTAMs, the schedule, and the SoC test length. However, the vTAMs are not completely defined by the scheduling tool. For instance, assuming vTAM$_a$ presented in Figure 4.5, vTAM$_a$ is defined as $\{--, --, w_3, \{22\}, \{21, 22, 32\}\}$ after the scheduling.

The FIFO depth $d$ and packet size $k$ are defined in the DfT optimization phase (Chapter 9 and (AMORY et al., 2007)) by a simulation based approach. This step takes the partial DfT defined in the scheduling and simulate the design to determine the minimal FIFO depth and minimal packet size for each vTAM, minimizing the silicon area. After this step all parameters required that represent the proposed test architecture are defined and the HDL of the DfT modules can be generated.

## 4.9 Comparison with Previous Approaches

Despite of the problem statement previously presented in Section 3.3, the proposed problem statement requires the graph $G$ to inform how the modules are connected to each other (while the previous algorithm only needs the set of modules). It also requires input and output ports for each module, the function $r()$ representing the NoC routing algorithm, and the channel width $c$.

On the other hand, compared to previous NoC-reuse approaches (COTA; LIU, 2006), our approach only requires the channel width and the routing algorithm. The previous approach requires full cycle-accurate time model to determine when the resources are available.

In conclusion, the proposed approach is between the conventional and the previous NoC-reuse test scheduling in terms of the amount of information required. In this sense, our test model is more general than (COTA; LIU, 2006).

This section also analyze some other features of the proposed approach, comparing it informally to dedicated TAM (presented in Section 2.1, page 35, and illustrated in Figure 2.2). The results presented along this thesis confirm these claims.

Unlike stated in previous NoC reuse approaches (COTA et al., 2003), the silicon area required to implement the DfT for NoC-reuse test architectures in this thesis is higher than the silicon area for test architectures based on dedicated TAMs. On one hand, the extra silicon area consists of just gate logic, not long wires as dedicated

[2] In most actual NoCs it is not possible to sustain the full physical channel bandwidth since there is also other network control information in the traffic. For this reason we assume that the maximal vTAM bandwidth is half of the physical channel bandwidth.

TAMs. As presented before, long wires are expected to cost more than logic in terms of design effort, delay, and power dissipation. On the other hand, it does not mean that the proposed DfT is less efficient than previous one. It means that we take into account design issues not considered before. For instance, *we have proposed ATE interfaces for the test pins that are not connected to the NoC and test wrappers that deals with functional protocol.* None of these required DfT has been addressed before.

In terms of *timing for test data transportation*, both proposed and conventional SoC test are based in constant and continuous test data streaming which is naturally provided by dedicated TAMs. The previous approaches based on functional interconnect reuse usually do not provide this feature (COTA et al., 2003; HWANG; ABRAHAM, 2001; HARROD, 1999).

For the same reason, *the proposed model might have higher latency from the test pins to the CUT.* We say "might" because a network design can be optimized, for example, for latency. For instance, packet switching networks have typically more latency than circuit switching networks once the circuit has been established. The latency for a circuit switching network can be one clock cycle per hop. However, even if a network with more latency is used, latency is still not important for a test model.

Consider the general formula of data transportation time is $t = l + \frac{v}{b}$, where $t$, $l$, $v$, and $b$ denote, respectively, time for transportation, latency of the medium, data volume, and bandwidth (MCCABE, 2003). The latency of dedicated TAMs consists of few buffers and typically results in few units of clock cycles. However, as stated in the introduction, the functional interconnects are evolving to segmented wiring to handle DSM effects on long wires, thus, the latency of functional interconnect is increasing. For example, a path in a NoC consists of several modules (two network interfaces and several routers). Therefore, on one hand, it can be concluded that the latency of dedicated TAMs is typically smaller than the latency of functional interconnects. On the other hand, the impact of the latency on the core test length is minimal since $l \ll \frac{v}{b}$. The typical test data volume of a core can be around hundreds of thousands or millions of bits. Thus, *latency can be neglected no matter if the latency is 3 clock cycles for dedicated TAMs or 30 for NoCs.* In addition, interconnects support pipelining to partially hide this latency, thus, only the very first word of data feels the latency effect since the proposed approach is based on non-preemptive test. For these reasons we concentrate in the minimization of $\frac{v}{b}$ to reduce the core test length. However, the latency is still important for the test application because the tester needs to know the exact time data is inserted and evaluated into the chip. The exact latency for a given interconnect can be measured, for example, by using logic simulation and counting the number of clock cycles required for the first word to go from the ATE to the CUT [3]. Realizing that latency can be taken off the calculation helps the interconnect abstraction (i.e. the pipeline abstraction mentioned before).

[3] Recall that this simulation of test patterns is not an additional burden in the design, but it is a required step to validate the generated test patterns. This is the reason why most ATPG tools automatically generate a test bench to ease this step. What we propose is just to take this latency information from the simulation, not from the test model.

*The wrapper of the proposed model does not need bypass mode in most cases.* The bypass mode is required when a core $a$ is not connected directly in the NoC, but is connected to another core $b$ that is connected to the network (see case three in Figure 2.12 in page 45 for an illustrative example). So, the wrapper of core $b$ must have bypass mode to indirectly connect core $a$ to the NoC.

*The optimization problem of dedicated TAM and the proposed model also seem to be similar.* Figure 4.6 compares both test architectures: (*i*) the conventional approach distributes the total number of test wires into independent TAMs. The proposed approach distribute the test pins among the test partitions, where the ATE interface connects the pins to the network. (*ii*) all cores of a dedicated TAM are tested using the same constant test bandwidth (i.e. number of test wires), likewise a NoC partition. Pipelines of different partitions have different width. (*iii*) since dedicated TAMs are designed only for test, they naturally support continuous and constant data flow with zero-jitter. The DfT modules (wrappers and ATE interface) of the proposed approach also provide the same communication pattern in an end-to-end manner (i.e. from the test pins to the CUT). (*iv*) TAMs of the conventional approach are independent from other TAMs and the cores within a TAM are tested sequentially. The NoC partitions of the proposed approach are independent from the other partitions and the cores are also tested sequentially. The use of algorithms previously proposed for test architecture with dedicated TAMs (GOEL; MARINISSEN, 2003a; MARINISSEN; GOEL; LOUSBERG, 2000) for NoC reuse also demonstrates that both problems can be similar.

Despite of all these similarities there are also some differences. The conventional approach typically has no constraint regarding the order the cores are tested within a TAM like in the proposed model. In addition, the proposed approach requires neighborhood and TAM width constraints, presented in Chapter 8.

## 4.10 Summary

This chapter presented an overview of the proposed system test model for BE NoC-based chips. The model divides the network in partitions, where each partition is tested independently from each other, avoiding interference and packet collisions that delay a message. Thus, complex, cycle-accurate, and dynamic network models that model packet collision and congestion are not required for test scheduling. For this reason, the proposed model is simpler and more general than previous NoC-reuse schemes based on packet scheduling. The proposed model is built on top of the DfT modules (wrappers and ATE interface), designed to reinforce the compatibility with dedicated TAMs and the external tester timing features. In addition, the proposed DfT abstracts the NoC internals as simple pipelines with scalable width.

As result, the combination of the proposed DfT modules and the NoC partition test model gives the support to the six features described in the beginning of this chapter: (1) *general test approach* that works for a wide range of networks since few functional features are required; (2) *compatibility* with conventional test since the tester timing features are respected; (3) it enables the *trade-off of test costs* as dedicated TAMs since the pipeline width is scalable; (4) the DfT modules are based on well-defined interfaces, then the DfT modules are *easily inserted in the chip design*; (5) the proposed method does not modify the network, *minimizing the*

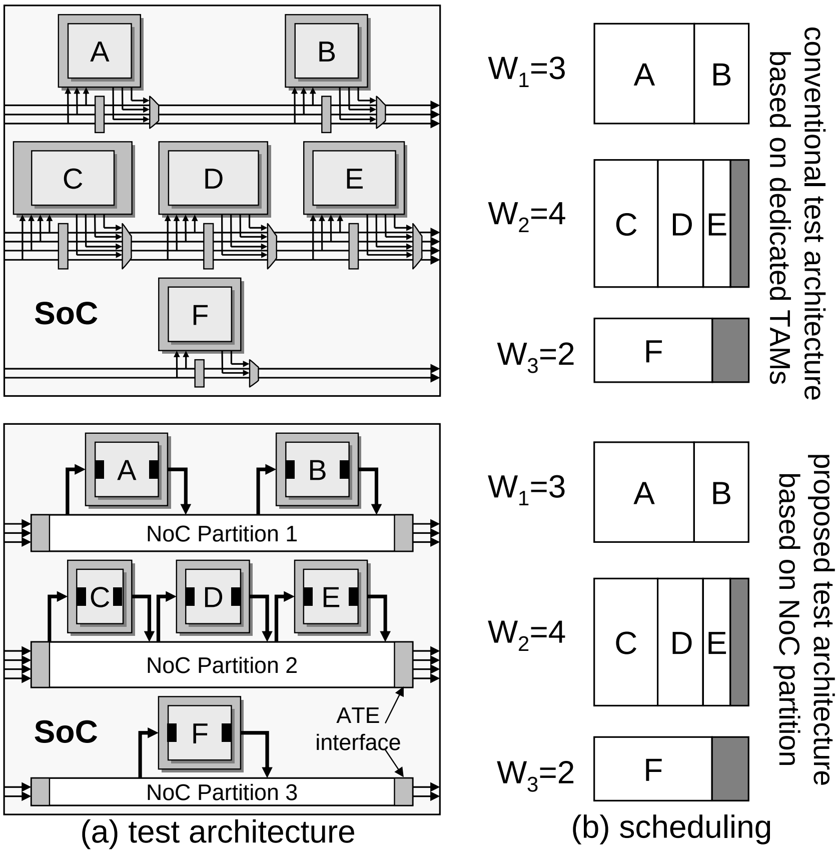


Figure 4.6: Comparing conventional and proposed test architectures. The gray area represents the DfT modules (test wrappers and ATE interfaces). The black rectangles in the cores represent the ports.

*influence on the functional domain*; and finally, (6) the concurrent NoC partitions *test cores in parallel* to reduce the core test.

# 5 TEST WRAPPERS FOR CORES

This chapter proposes a new core test wrapper design approach which transports streaming test data into and out of an embedded core via its functional data ports. These ports are typically based on standardized protocols such as AXI, DTL, and OCP. Our new wrapper design allows functional interconnect, such as an on-chip bus or NoC, to transport test data to embedded cores, and hence eliminates the need for a conventional dedicated TAM, such as a TestRail or test bus. Our approach leaves the tester, the embedded core, and its test unchanged, while the functional interconnect can handle the test data transport as a regular data application. The proposed wrapper requires guaranteed throughput and zero latency variation. We show how to use the wrapper over functional interconnects with and also without support to guaranteed services.

For 672 example cases based on the ITC'02 SoC Test Benchmarks, our new approach in comparison to the conventional approach shows an average wrapper area increase of 14.5%, which is negligible at SoC level, especially since the dedicated TAM can be eliminated. Our new approach furthermore decreases the core test length on average with 3.8%.

This chapter is organized as follows. Section 5.1 presents the problem statement. Section 5.2 presents an overview of the wrapper design. Section 5.3 shows the wrapper design and optimization procedure in details. Section 5.4 shows experimental results. Finally, Sections 5.5 and 5.6 presents an overall discussion and conclusion, respectively.

## 5.1 Problem Statement

The core must use exactly one protocol port as test input and exactly one other protocol port as test output. As not all bidirectional ports support full-duplex communication, read-write ports are used in only one direction, in order to allow scan-in and scan-out operations during test to overlap in time. Consequently, the core need to have two protocol ports: one to serve as test input $i^p$ and one to serve as test output port $o^p$. We assume that at least two suitable ports are available. The terminals of these ports must be classified according to Section 5.3.1 and we need the behavior of the protocol and the rule of each terminal in the protocol.

In addition, all the test information required for the conventional wrapper design is also required, like

- number of test patterns $p_g$;
- number of functional input terminals $i_g$;

- number of functional output terminals $o_g$;
- number of functional bidirectional terminals $b_g$;
- the number of scan chains $s_g$ for each scan chain $k$ with scan length $l_{g,k}$ (for hard cores only);
- the number of scan flip-flops $f_g$ (for soft cores only).

Finally, we need to determine the design of a core test wrapper, which enables the CUT to be tested by the ATE via the NoC without modifying the ATE, the interconnect, the CUT, or its test. The design of the wrapper is optimized such that the CUT's test length and wrapper silicon area are minimized.

## 5.2 Wrapper Design Overview

Figure 5.1 shows both the conventional (a) and the new (b) IEEE Std. 1500 (SILVA, 2005; SILVA; MCLAURIN; WAAYERS, 2006) compliant wrapper designs, and illustrates their differences. In the InTest mode of the conventional wrapper, access of all functional inputs and outputs to the core is intercepted by the Wrapper Boundary Register (WBR), while ports to dedicated TAMs provide test access. In the new wrapper design, there is no dedicated TAM. Test access to the core is provided via a subset of the functional inputs and outputs, viz. via one selected input port and one selected output port. Functional inputs and outputs that do not belong to the selected ports are intercepted by the WBR.

The algorithm to design and optimize the test wrapper consists of the following three steps.

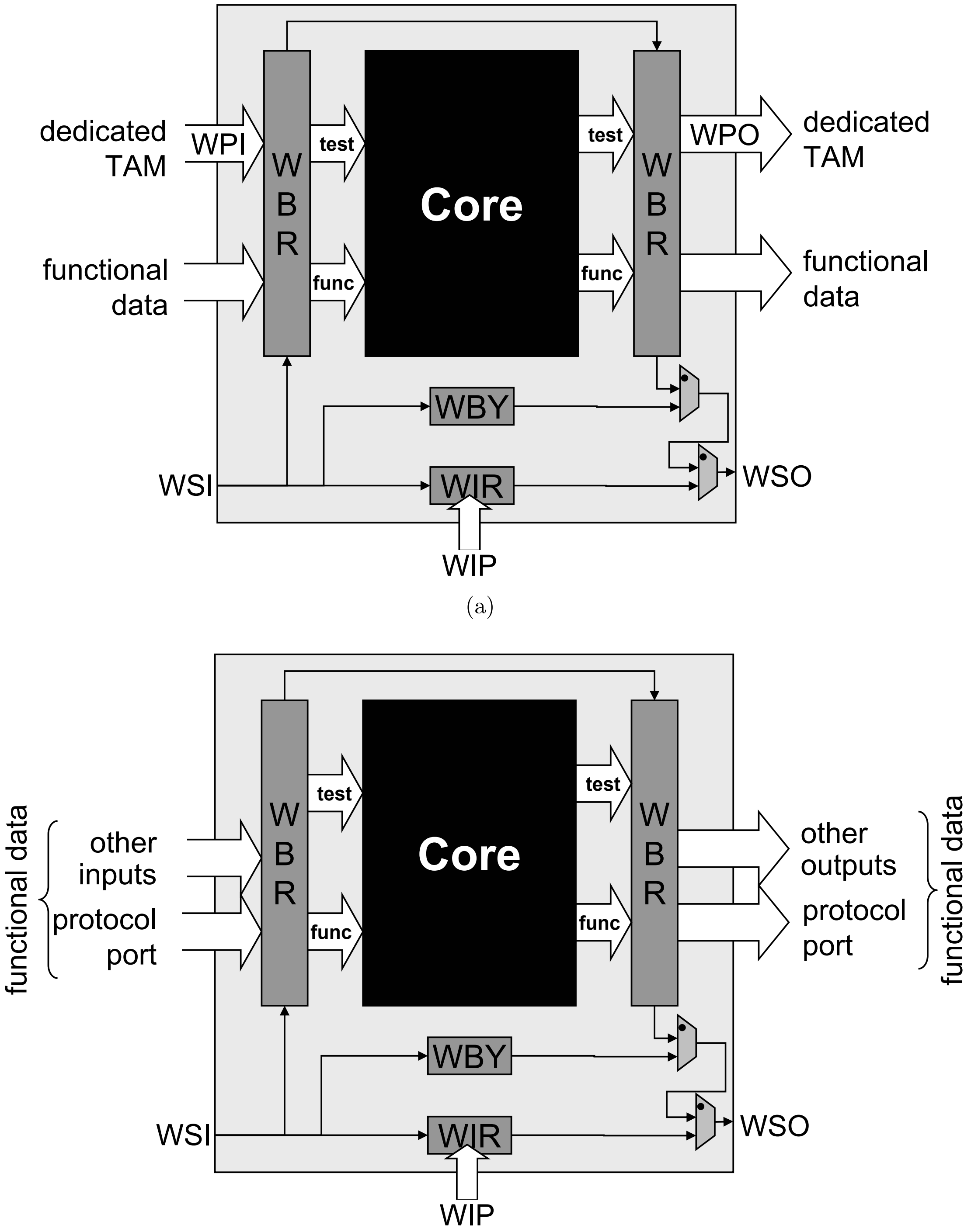
1. *Calculation of parallel-to-serial loading and serial-to-parallel unloading characteristics.*
   Stimulus bits arrive strictly periodically in words of $c_i$ bits at the selected test input port $i$. There, they are equally divided over the $w$ wrapper chains and serially shifted into the wrapper and core. This process is repeated every $p_i$ clock cycles, with

   $$p_i = \left\lfloor \frac{c_i}{w} \right\rfloor \tag{5.1}$$

   Note that of the $c_i$ data bits in each parallel word, only $\lfloor \frac{c_i}{w} \rfloor \times w$ are actually used to carry stimulus bits; the remaining ($c_i$ mod $w$) bits carry unused data, and consequently will be assigned special wrapper cells in Step 4. The response side is handled likewise, where every $p_o = \lfloor \frac{c_o}{w} \rfloor$ clock cycles, $p_o$ response bits are unloaded in parallel from the test output port $o$ of each of the $w$ wrapper chains [1].

2. *Core terminal classification.*
   Based on their role in the test operation, core terminals are classified accorting to: ($i$) the type of wrapper cell for the terminal, ($ii$) the position of the wrapper

[1] we might use $c$ when $c_i = c_o = c$, i.e. when the port input data width, the port output data width, and the channel width are equal.



(a)

(b)

Figure 5.1: Overview of the (a) conventional and (b) new IEEE Std. 1500 compliant wrapper design.

cell in a wrapper scan chain, and (*iii*) the control signals to the wrapper cell. The core terminal classification is described in more detail in Section 5.3.1.

3. *Actual wrapper design.*
   Based on the outcome of the core terminal classification, the wrapper is instantiated, according to the following sub-steps.

   (a) Assign wrapper cells to individual core terminals, as described in Section 5.3.2.

   (b) Partition and order the wrapper cells and core-internal scan chains over the $w$ wrapper chains, as described in Section 5.3.3.

   (c) Connect the control logic to the various wrapper cells.

## 5.3 Wrapper Design and Optimization

### 5.3.1 Core Terminal Classification

Conventional wrapper design (MARINISSEN; GOEL; LOUSBERG, 2000) distinguishes four classes of core terminals: functional inputs (FI), functional outputs (FO), scan inputs (SI), and scan outputs (SO). In the new approach, in which test data reaches the core via reused functional ports, we add eight new classes, for the terminals of the functional ports.

- SDI, SDO: Selected Data Inputs and Outputs, i.e., the data terminals of the functional port that has been selected to serve as test input (output), and carry actual stimulus (response) data.
- RSDI, RSDO: Remaining Selected Data Inputs and Outputs, i.e., those ($c_i$ mod $w$) data bits of the selected test input port $i$ and ($c_o$ mod $w$) data bits of the selected test output port $o$ that carry unused data.
- DI, DO: Data Inputs and Outputs, i.e., the data terminals of the remaining functional ports not selected to carry test data.
- CI, CO: Control Inputs and Outputs, i.e., the non-data terminals of all functional ports.
- FI, FO: Functional Inputs and Outputs, i.e., all functional terminals that are not part of the selected test ports.
- SI, SO: Scan Inputs and Outputs.

The twelve classes provide a complete partitioning for all digital data terminals of the core, i.e., all classes are mutually exclusive and their union equals all terminals [2].

The classification of the functional port terminals of the example in Figure 5.2 is shown in Table 5.1. For both ports, the `dtl_wr_data[31:0]` terminals are selected as test input (output) ports and hence are classified as SDI and SDO respectively. Assuming $w = 3$, both selected ports have ($c \bmod w$) = ($32 \bmod 3$) = 2 unused

[2] we might use $DI_{in}$, $DO_{in}$, $CI_{in}$, and $CO_{in}$ to refer, for example, to the DI terminals of the input port. $DI_{out}$, $DO_{out}$, $CI_{out}$, and $CO_{out}$ refer to the sets of the output port

terminals, say `dtl_wr_data[31:30]`, which are moved into the RSDI and RSDO classes. The `dtl_rd_data[31:0]` terminals are *not* selected as test input and output ports, and hence are left as DI and DO. The sets CI and CO include all control terminals, i.e., all `cmd` signal group terminals and the `valid` and `accept` signals of the `write` and `read` signal groups.

Table 5.1: Example classification of port terminals.

| **Class** | **Test Input Port** | **Test Output Port** |
|---|---|---|
| SDI | `dtl_wr_data[29:0]` | - |
| RSDI | `dtl_wr_data[31:30]` | - |
| SDO | - | `dtl_wr_data[29:0]` |
| RSDO | - | `dtl_wr_data[31:30]` |
| DI | - | `dtl_rd_data[31:0]` |
| DO | `dtl_rd_data[31:0]` | - |
| CI | `dtl_cmd_valid,`<br>`dtl_cmd_read, dtl_rd_valid,`<br>`dtl_wr_valid, dtl_rd_accept,`<br>`dtl_cmd_blocksize[5:0],`<br>`dtl_cmd_addr[31:0]` | `dtl_cmd_accept,`<br>`dtl_wr_accept` |
| CO | `dtl_cmd_accept,`<br>`dtl_wr_accept` | `dtl_cmd_valid, dtl_rd_valid,`<br>`dtl_cmd_addr[31:0],`<br>`dtl_cmd_blocksize[5:0],`<br>`dtl_wr_valid,`<br>`dtl_rd_accept,`<br>`dtl_cmd_read` |

### 5.3.2 Wrapper Cell Design and Assignment

In our wrapper, we use two types of IEEE Std. 1500 compliant wrapper cells. All terminals use a 'regular' wrapper cell, except for the CO-type terminals, which use a special variant of the 'regular' wrapper cell. Any IEEE Std. 1500 compliant wrapper cell can serve as 'regular' wrapper cell. We prefer IEEE Std. 1500 wrapper cell WC_SD1_COI, as depicted in Figure 5.3(a). This wrapper cell is small (one flip-flop only) and the activation of both InTest and ExTest modes guarantees full test coverage of the wrapper cell itself, as the captured signal is tapped off *after* the functional multiplexer.

The CO-type core terminals require a special variant of the 'regular' wrapper cell, because they need to assure that, as far as the on-chip bus or NoC is concerned, also in test mode the functional port protocol is handled as normal, such that the transport of test data is not interrupted. For example: the `valid/accept` handshakes of the command, write, and read groups of a DTL port need to be completed as usual. The special wrapper cell for CO-type terminals consists of a generic part and a terminal-specific part. The generic part is a *guarded* variant of the 'regular' wrapper cell. For our preferred wrapper cell WC_SD1_COI, the guarded variant is IEEE Std. 1500 wrapper cell WC_SD1_COI_G as depicted in Figure 5.3(b). Its functional output can be set to the value of signal `prot_value`. The circuitry that generates `prot_value` is specific for each CO terminal.

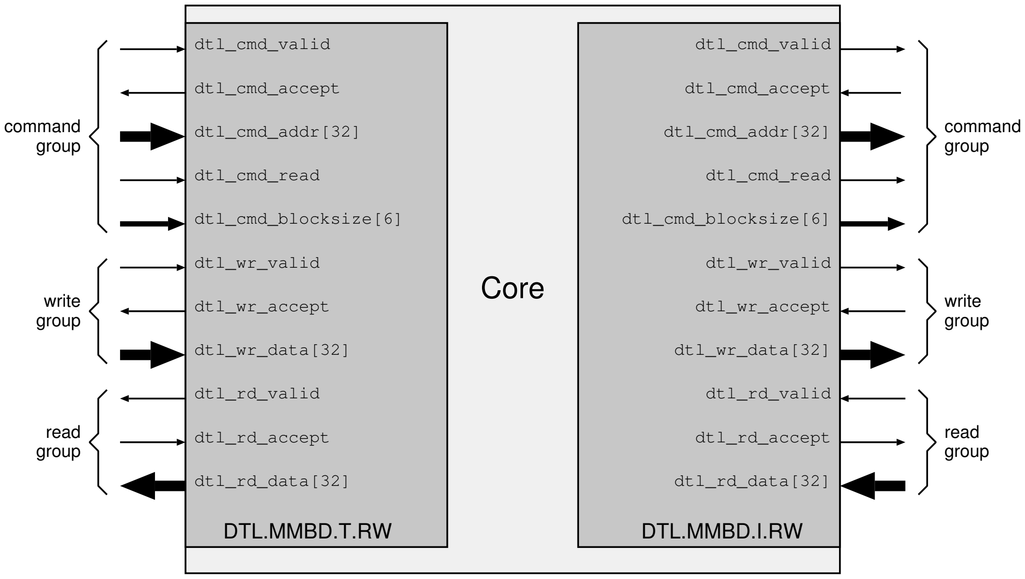


Figure 5.2: Example core with two DTL read-write ports.

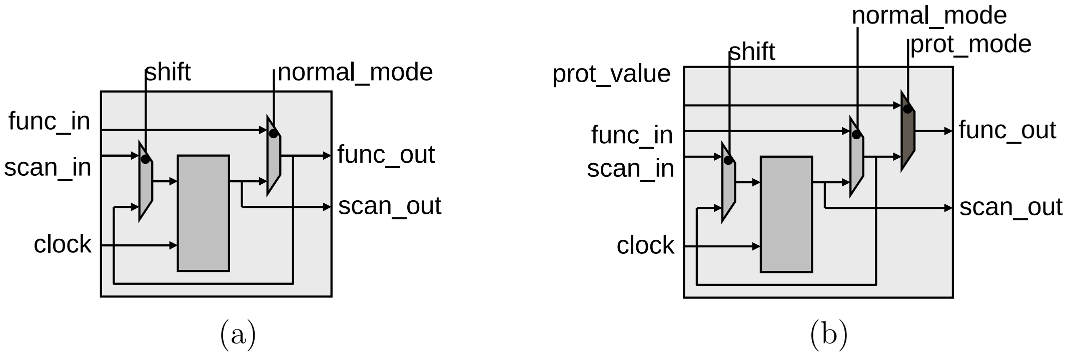


Figure 5.3: Implementation of wrapper cells (a) WC_SD1_COI and (b) WC_SD1_COI_G.

The generic version of the control generator has a periodic output behavior, which is characterized by a binary-coded output value $v$ and a period $p$. For $p-1$ consecutive clock cycles, the output value is $\overline{v}$, while for the next clock cycle, the output is $v$. The hardware implementation uses a simple counter. Some CO terminals require hard-coded '0' or '1' signals; this is captured by $(v,p)=(0,1)$ and $(v,p)=(1,1)$ respectively. The control generator implementation for these signals uses tie-off cells.

Table 5.2 describes for the selected test output port of our example in Figure 5.2 which output signals are required on its CO-type terminals in order to keep the functional port protocol running. The port is used as test output via its `dtl_wr_data` terminals, and hence `dtl_cmd_read` is kept at a hard-coded '0'. Every $p_o$ clock cycles a new word with test responses is available for parallel read-out from the scan chains. To write this word to the interconnect `dtl_wr_valid` is '1' every $p_o$ clock cycles. The specified bandwidth $bout(o)$ is achieved for a given block size $s(o)$. Hence the binary six-bit equivalent of $s(o)$ is output on `dtl_cmd_blocksize[6]`. Every $(s(o)\times p_o)$ cycles, a block of $s(o)$ words has been output, and this initiator port should issue a new command, in order to instruct the NoC to accept another block. Terminals `dtl_cmd_addr[31:0]` and `dtl_rd_accept` are not used, and hence kept at hard-coded ‘0’.

Table 5.2: Control generator parameters for test output DTL port.

| **Terminal** | **Param.** $(v, p)$ | **Description** |
|---|---|---|
| `dtl_cmd_valid` | $(1, s(o) \times p_o)$ | every $s(o)$ words, a new command is issued |
| `dtl_cmd_addr[31:0]` | ‘0’ $= (0, 1)$ | the address is not used and hence kept at ‘0’ |
| `dtl_cmd_read` | ‘0’ $= (0, 1)$ | only the *write* side of this port is used and hence read=0 |
| `dtl_cmd_blocksize[6]` | $(s(o), 1)$ | the block size is $s(o)$ |
| `dtl_wr_valid` | $(1, p_o)$ | every $p_o$ clock cycles, a new word with test responses is available |
| `dtl_rd_accept` | ‘0’ $= (0, 1)$ | the *read* side of this port is not used and hence accept=0 |

### 5.3.3 Partitioning and Ordering of Wrapper Chain Items

Wrapper chains are made up from wrapper cells and core-internal scan chains. We need to construct $w$ wrapper chains, but typically there are many more wrapper cells and core-internal scan chains than $w$ wrapper chain items. Hence, we need to partition the wrapper cells and core-internal scan chains over the $w$ wrapper chains, and subsequently determine the order of the items per wrapper chain. Our approach to do this is a modification of the conventional wrapper design optimization algorithm of (MARINISSEN; GOEL; LOUSBERG, 2000).

In conventional wrapper design, the test length $T_{conv}$ for a wrapped core is defined as

$$T_{conv} = (1 + \max(s_i, s_o)) \times p_g + \min(s_i, s_o) \tag{5.2}$$

where $p_g$ denotes the number of test patterns, and $s_i$ and $s_o$ denote the scan-in and scan-out length for the wrapped core, respectively (MARINISSEN; GOEL; LOUSBERG, 2000). To minimize $T_{conv}$, both $s_i$ and $s_o$ must be minimized. All wrapper input cells and all core-internal scan chains participate in a scan-in operation and hence might contribute to $s_i$; likewise, all wrapper output cells and core-internal scan chains participate in a scan-out operation and therefore might contribute to $s_o$. The conventional partitioning algorithm in (MARINISSEN; GOEL; LOUSBERG, 2000) first addresses the $\mathcal{NP}$-hard problem of partitioning the core-internal scan chains over the available number of to-be-constructed wrapper chains, before partitioning the wrapper input and output cells. This approach aims to minimize both $s_i$ and $s_o$. The subsequent step of ordering of wrapper items per wrapper chain is done such that (1) the wrapper input cells are followed by (2) the core-internal scan chains, which in turn are followed by (3) the wrapper output cells. A schematic view of the resulting conventional wrapper chain is depicted in Figure 5.4(a).

A schematic view of the wrapper chains resulting from our new wrapper design approach is shown in Figure 5.4(b). The main differences are formed by the category of SDI wrapper input cells, through which stimuli are loaded into the wrapper chain

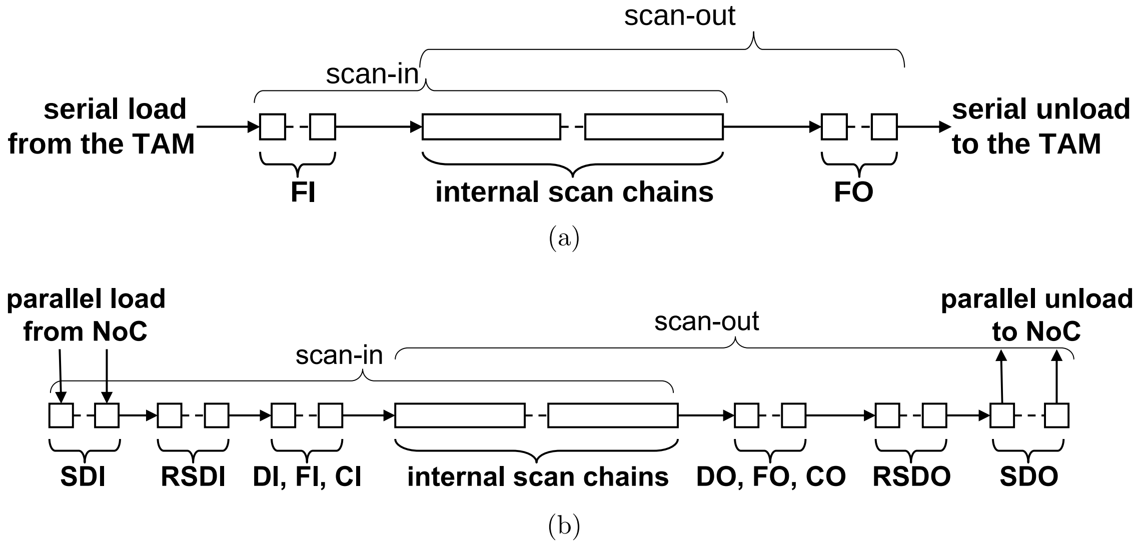


Figure 5.4: Schematic view of conventional (a) and new (b) ordering of wrapper items per wrapper chain.

in a parallel fashion, and the category of SDO wrapper output cells, through which responses are unloaded from the wrapper chain in a parallel fashion.

- The test stimuli of a test pattern arrive at regular intervals of $p_i$ clock cycles at the SDI wrapper input cells. All of them are shifted into the wrapper chains, apart from the last word, that can be directly consumed in parallel for testing. Similarly, for each test pattern the first word of responses is directly transported away in parallel from the SDO wrapper output cells, after which the subsequent words are shifted out of the wrapper chains at regular intervals of $p_o$ clock cycles. Hence, the test length $T_{new}$ is redefined as

$$T_{new} = (1 + \max(t_i, t_o)) \times p_g + \min(t_i, t_o) \tag{5.3}$$

with

$$t_i = \left( \left\lceil \frac{s_i}{p_i} \right\rceil - 1 \right) \times p_i + 1 \tag{5.4}$$

$$t_o = \left( \left\lceil \frac{s_o}{p_o} \right\rceil - 1 \right) \times p_o + 1 \tag{5.5}$$

- In order to minimize test length $T_{new}$, the wrapper item should be *partitioned* such that both all SDI cells and all SDO cells are evenly distributed over the $w$ to-be-constructed wrapper chains.

- For wrapper item *ordering*, it is important that all SDI cells are placed at the start of the wrapper chain, as they are the entry point of stimuli into the wrapper chain, and otherwise some wrapper chain items are unreachable for scan access. Similarly, all SDO cells should be placed at the end of the wrapper chain.

Algorithmically, the partitioning of wrapper items is done in a five-step approach, which is clearly based on the three-step approach in (MARINISSEN; GOEL; LOUSBERG, 2000).

1. Assign the core-internal scan chains to the $w$ wrapper chains, such that the maximum sum of scan lengths assigned to a wrapper chain is minimized. Algorithms like LPT and COMBINE can be used (MARINISSEN; GOEL; LOUSBERG, 2000). The resulting partition is named $\mathcal{P}_s$.

2. Assign the wrapper input cells in $RSDI \cup DI \cup CI \cup FI$ to the $w$ wrapper chains on top of $\mathcal{P}_s$, such that the maximum scan-in length of all wrapper chains is minimized. The resulting partition is named $\mathcal{P}_w^{in}$.

3. Assign the wrapper input cells in $SDI$ to the $w$ wrapper chains on top of $\mathcal{P}_w^{in}$. The resulting partition is named $\mathcal{P}^{in}$.

4. Assign the wrapper output cells in $RSDO \cup DO \cup CO \cup FO$ to the $w$ wrapper chains on top of $\mathcal{P}_s$, such that the maximum scan-out length of all wrapper chains is minimized. The resulting partition is named $\mathcal{P}_w^{out}$.

5. Assign the wrapper output cells in $SDO$ to the $w$ wrapper chains on top of $\mathcal{P}_w^{out}$. The resulting partition is named $\mathcal{P}^{out}$.

### 5.3.4 NoCs With and Without Guaranteed Services

In fact, the main difference between NoC with and without guarantees is that NoCs with guarantees already provide a FIFO at the core input that can be used to eliminate load fluctuation. These FIFOs are inside the NI, hidden from the user (RADULESCU et al., 2005). Another impact in the wrapper design is that, since they are inside the NI, they cannot be modified according to the test needs. So, to deal with a given maximal bandwidth, we have to add two more steps before the step of the procedure presented in Section 5.2.

1. *Selection of test input and output ports.*
   In case multiple ports are available that could serve as test input or test output port, exactly one test input port and exactly one test output port are selected. To minimize the resulting test length, we select two disjunct ports $i$ and $o$ (with $i \neq o$) of the CUT connected to the ATE such that the resulting test bandwidth $b_{test}$ is maximized, with

$$b_{test} = \min(b_i^{in}, b_o^{out}) \quad . \tag{5.6}$$

   According to Equation (5.6), the test bandwidth is determined by the minimum of the bandwidths of the selected input and output port, as the bandwidths of wrapper scan input and output needs to be equal.

2. *Calculation of the number of wrapper chains.*
   At the core's test frequency $f$, the functional interconnect can deliver at most $\left\lfloor \frac{b_{test}}{f} \right\rfloor$ test stimuli bits per clock cycle via the selected test input port, and receive an equal amount of test response bits via the selected test output port. Hence, the wrapper should not contain more than $w^{max}$ wrapper chains with

$$w^{max} = \left\lfloor \frac{b_{test}}{f} \right\rfloor \tag{5.7}$$

   to use the given test bandwidth provided by the functional interconnect.

Some examples of NoCs that can give performance guarantees include Æthereal (GOOSSENS; DIELISSEN; RADULESCU, 2005), Mango (BJERREGAARD; SPARSO, 2005a), Nostrum (MILLBERG et al., 2004), and Sonics (WEBER et al., 2005).

The proposed wrapper design also works for NoCs without guarantees. These NoCs do not have FIFOs at the core inputs, thus they might be required in the wrapper. The advantage in this case is that the wrapper design is not constrained to $w^{max}$, but to $c$. On the other hand, the FIFO might be required, increasing the DfT silicon area. However, once the FIFOs will be used only for test, we can define the best (the sufficient) FIFO depth for each wrapper.

Some examples of NoCs that do not provide performance guarantees include RaSoC (ZEFERINO; SUSIN, 2003) and Hermes (MORAES et al., 2004).

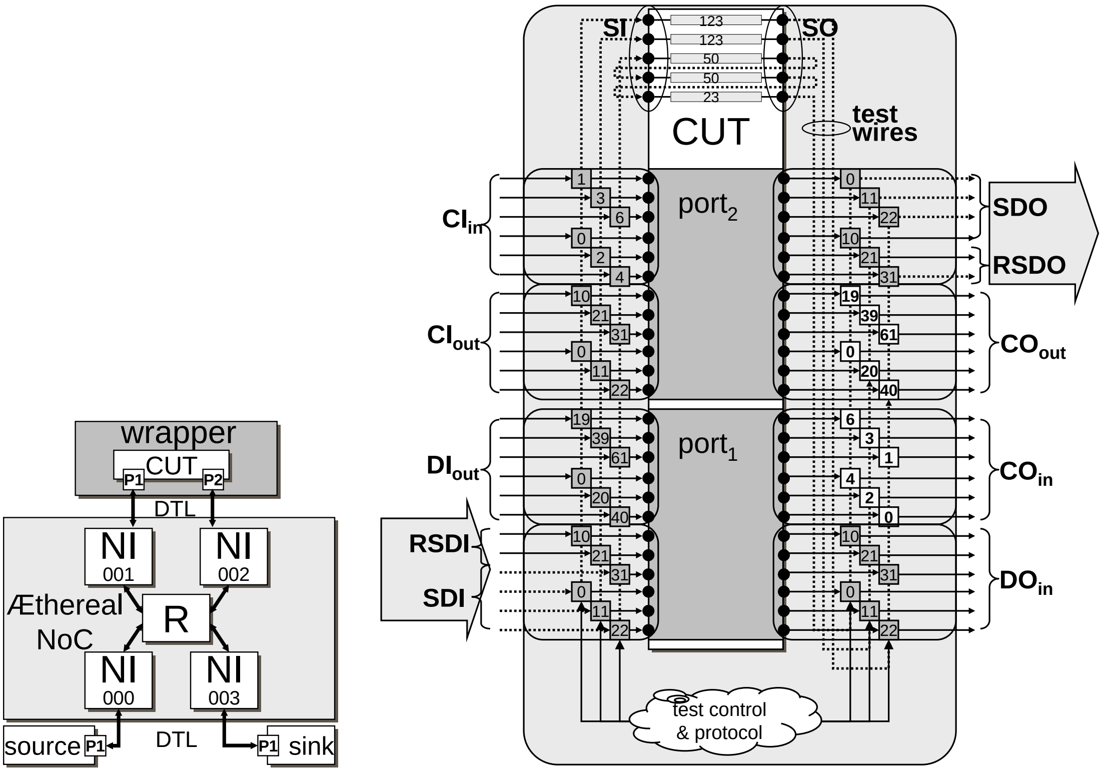
## 5.4 Experimental Results

### 5.4.1 Simplified Illustrative Example

We have implemented our wrapper design for a simplified illustrative example core and NoC. The NoC is an automatically generated, simple Æthereal network (GOOSSENS; DIELISSEN; RADULESCU, 2005), consisting of one router and four network interfaces, based on a 32-bits data DTL protocol (PHILIPS SEMICONDUCTORS, 2002). As depicted in Figure 5.5(a), two NoC ports connect to the respestive ATE source and sink, while the two other ports connect to the CUT.

The CUT has two DTL ports, Port 1 and Port 2, with 133 terminals each and functional data word width $c = 32$. Besides the two ports, the CUT has no other functional terminals. The CUT has five internal scan chains of lengths 123, 123, 50, 50, and 23 flip-flops, and $p_g = 10$. Port 1 is selected to receive stimuli and Port 2 is selected to send out responses. The test data flow for stimuli is from the source through the NoC into the CUT via Port 1, while test responses flow from CUT's Port 2 via the NoC into the sink. $b_i^{in} = 1600$ Mbits/s, $b_o^{out} = 2400$ Mbits/s, and $f = 500$ MHz, and, by Equation 5.7, we can afford a wrapper with $w = 3$ wrapper chains. According to Equation 5.1, test data arrives with period $p_i = p_o = 10$ clock cycles. For Port 1: |SDI|=30, |RSDI|=2, |DO|=32, |CI|=62, and |CO|=7. For Port 2: |SDO|=30, |RSDO|=2, |DI|=32, |CI|=7, and |CO|=62. The wrapper chain scan lengths are $s_i = s_o = 168$. The resulting wrapper design is shown in Figure 5.5(b). Based on Equation 5.3, test length $T_{new} = 1781$ clock cycles. Note that this represents a 4.1% test length reduction compared to a conventional wrapper design with three wrapper chains, which would have a test length $T_{conv} = 1858$ clock cycles. This reduction occurs because the last stimulus word of each pattern is loaded in parallel into the SDI wrapper cells and does not require any further shifting in, and likewise the first response word of each pattern is loaded in parallel into the SDO wrapper cells.

Our example was implemented in register-transfer-level VHDL and verified through simulation. The experiment showed a modest area increase for the new wrapper in comparison to a conventional wrapper. The number of equvalent gates required to implement all wrapper cells went from 2910 to 3000 (an increase of +3.1%), while 489 equivalent gates were required to implement the additional control logic to complete correct protocol operation. The total relative wrapper area increase was +19.9%. Note that wrappers are typically small compared to the overall SoC size,



(a) (b)

Figure 5.5: Illustrative example consisting of NoC and CUT (a) and the detailed wrapper design for the CUT (b).

and hence, this area impact is negligible at SoC level.

Figure 5.6 presents a waveform with some relevant signals of the system presented in Figure 5.5(a). The waveform is divided into four signal groups: the source, the cut target, the cut initiator, and sink signal groups.

The first signal group shows some relevant signals between the NoC and the source module, which represents the initiator ATE interface. It can be observed that data is sent at regular time intervals (every $p_i$ clock cycles). The second signal group shows signals between the NI and the input port of the wrapper. More specifically it shows signals right after the input FIFO (if this is required) since we can see that data is as regular as the data sent by the initiator ATE interface. The third and fourth signal groups represent the response path. The third signal group represents the data sent by the wrapper output port to the NoC. The response is received in the target ATE interface, the forth signal group.

### 5.4.2 Wrapper Area and Core Test Length Impact

In this section, we present wrapper area and core test length results for our new wrapper design approach in comparison with conventional wrapper design, obtained on a large subset of cores in the ITC'02 SoC Test Benchmarks (MARINISSEN; IYENGAR; CHAKRABARTY, 2002). A wrapper design and optimization tool as described in this chapter has been developed in C++; it uses the COMBINE wrapper design routine (MARINISSEN; GOEL; LOUSBERG, 2000) as basis. The tool calculates the required number of gate-equivalents to design the wrapper and the

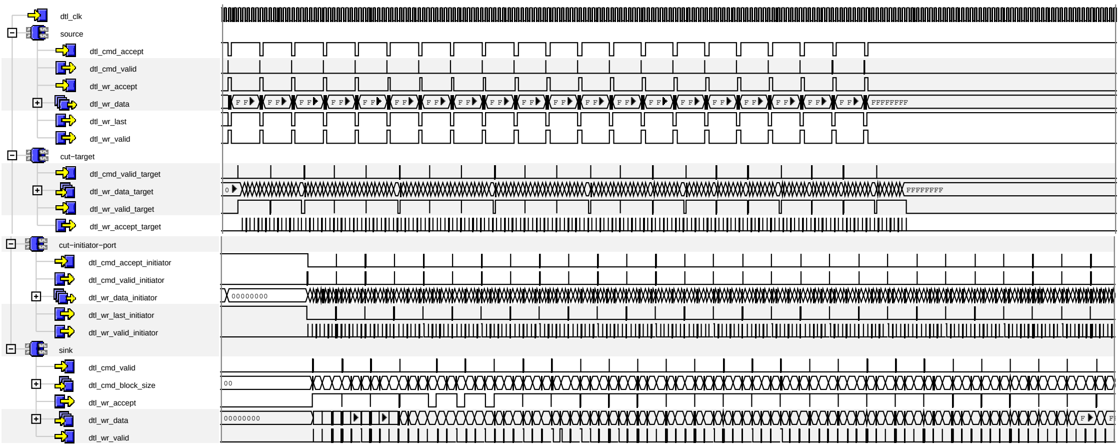


Figure 5.6: Wrapper waveform.

Table 5.3: List of considered ITC'02 SoC Test Benchmarks (MARINISSEN; GOEL; LOUSBERG, 2000) cores.

| **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 |

resulting test length in clock cycles. We do this for both the conventional wrapper design approach, which relies on a dedicated TAM, as well as for our new wrapper design approach, which reuses the functional protocol ports of the core.

The cores in the ITC'02 benchmarks do not have functional protocol ports (or, at least, they are not specified), while our method critically depends on their presence. Hence, we assume that every core has two DTL ports, of which one serves as test input and the other as test output. The assumed DTL test input port has 32 SDI terminals, 45 CI terminals, and 2 CO terminals. The assumed DTL test output port has 32 SDO terminals, 2 CI terminals, and 45 CO terminals. For the two DTL ports, each core needs 79 input terminals and 79 output terminals. Some of the 186 cores in the ITC'02 benchmarks do not have that many terminals, and hence are not considered in our evaluation. This leaves still 42 cores for our experiments, which are listed in Table 5.3.

We assume that bandwidths $b_i^{in}$ and $b_o^{out}$ and test frequency $f$ are such that the maximum affordable number of wrapper chains $w = 16$. As $c_i = c_o = 32$, $w = 16$ implies that $p_i = p_o = 2$. However, it is also possible to implement fewer, but longer wrapper chains, which means that we are not using the bandwidth to its maximum, and consequently accept a longer test length.

For all 42 cores considered and for a number of wrapper chains $w$ ranging from 1 to 16, we have calculated the numbers of gate-equivalents required to implement a wrapper and the corresponding test length in clock cycles, for both the conventional wrapper design algorithm (MARINISSEN; GOEL; LOUSBERG, 2000) and our new approach. In total, this involves $42 \times 16 \times 2 = 1344$ wrapper calculations. Figure 5.7 shows the average relative gate-equivalent count increase and the relative test length increase. The horizontal axes refer to cores with numbers in the same order they are listed in Table 5.3.

Figure 5.7(a) shows the increase in the number of gate-equivalents required to implement the new wrapper, relative to what was required for a conventional wrapper design. These numbers show only around 1% variation for varying values of $w$, and hence we have decided to show the average increase for $1 \leq w \leq 16$. The average wrapper area increase over all 672 cases is 14.5%, which is negligible at SoC level. Figure 5.7(b) shows that the test length impact of the new approach varies between a $-27.4\%$ and $+7.2\%$, with on average over all 672 cases a test length decrease of 3.8%.

### 5.4.3 Wrapper Area for NoCs Without Guaranteed Services

The only difference in the design of NoC with and without guaranteed services is that later might require a FIFO in the wrapper. It affects the silicon area of the wrapper, but not the test length. For this reason we show additional silicon area results with emphasis on the size of the FIFO.

The FIFO depth of a wrapper depends on the number of test wires assigned for the wrapper and it also depends on the physical channel width. Table 5.4 shows the size of the FIFO for different number of test wires and channel width. The method used to determine the minimal FIFO depth and packet size is presented in Chapter 9. These results say that the size of the wrapper for NoCs without guaranteed services can increase 0% to 44%.

## 5.5 Discussion

This section discusses the different results of core test length and silicon area of the proposed wrapper compared to the conventional wrapper design (GOEL; MARINISSEN, 2003a).

The extra silicon area required by our approach is due to the protocol logic and the FIFO (in case of NoCs without guaranteed services).

The difference in core test length is not so obvious as silicon area. According to Figure 5.7(b), there are some cases that the proposed approach is faster and other cases that it is slower than the conventional wrapper. The rest of this section explains why this variation might occur.

Let us give an example why the proposed approach is slower than the conventional wrapper. Table 5.5 shows the core test length as function of the assigned test wires for the illustrative example in Figure 5.5. The columns show the number of test wires, the test length for the proposed wrapper, the test length for (GOEL; MARINISSEN, 2003a), and the test length increase.

Different test length between our approach and (GOEL; MARINISSEN, 2003a)

Table 5.4: Size of the wrapper's FIFO in NoCs without guaranteed services.

| channel width | test wire | packet length | FIFO words | wrapper area (eq. gates) | relative FIFO area (%) |
|---|---|---|---|---|---|
| 8 | 1 | 2 | 0 | 1023 | 0 |
| | 2 | 5 | 2 | 1290 | 21 |
| | 4 | 14 | 5 | 1566 | 37 |
| 16 | 1 | 1 | 0 | 1558 | 0 |
| | 2 | 2 | 1 | 1696 | 5 |
| | 3 | 4 | 2 | 2043 | 25 |
| | 4 | 5 | 2 | 2033 | 25 |
| | 5 | 7 | 3 | 2158 | 30 |
| | 8 | 14 | 5 | 2512 | 40 |
| 32 | 1 | 1 | 0 | 2593 | 0 |
| | 2 | 1 | 0 | 2542 | 0 |
| | 3 | 2 | 0 | 2540 | 0 |
| | 4 | 2 | 1 | 2846 | 5 |
| | 5 | 3 | 1 | 2858 | 5 |
| | 6 | 4 | 2 | 3523 | 27 |
| | 8 | 5 | 2 | 3513 | 27 |
| | 10 | 7 | 3 | 3726 | 32 |
| | 16 | 14 | 5 | 4362 | 42 |
| 64 | 1 | 1 | 0 | 4615 | 0 |
| | 2 | 1 | 0 | 4561 | 0 |
| | 3 | 1 | 0 | 4548 | 0 |
| | 4 | 1 | 0 | 4510 | 0 |
| | 5 | 2 | 0 | 4509 | 0 |
| | 6 | 2 | 0 | 4509 | 0 |
| | 7 | 2 | 1 | 5148 | 6 |
| | 8 | 2 | 1 | 5141 | 6 |
| | 9 | 3 | 1 | 5144 | 6 |
| | 10 | 3 | 1 | 5153 | 6 |
| | 12 | 4 | 2 | 6491 | 29 |
| | 16 | 5 | 2 | 6481 | 29 |
| | 21 | 7 | 3 | 6893 | 34 |
| | 32 | 14 | 5 | 8092 | 44 |

Table 5.5: The core test length as function of the number of test wires.

| test wires | our test time | (GOEL; MARINISSEN, 2003a) test length | additional test length (%) |
|---|---|---|---|
| 1 | 5532 | 5532 | 0 |
| 2 | 2771 | 2771 | 0 |
| 3 | 1858 | 1858 | 0 |
| 4 | 1451 | *1396* | 3.9 |
| 5 | 1429 | *1363* | 4.8 |
| 6 | 1418 | *1363* | 4.0 |

Table 5.6: Test wires for our approach and (GOEL; MARINISSEN, 2003a). $DI_{in} = SDI \cup RSDI$ and $DO_{out} = SDO \cup RSDO$.

| test wire | $DI_{in}$ | $CI_{in}+$ $DI_{out}+$ $CI_{out}$ | scan chains | $DO_{in}+$ $CO_{in}+$ $CO_{out}$ | $DO_{out}$ |
|---|---|---|---|---|---|
| `tw[0]` | 8 | 0 | {123} | 0 | 8 |
| `tw[1]` | 8 | 0 | {123} | 0 | 8 |
| `tw[2]` | 8 | 43 | {50,23} | 43 | 8 |
| `tw[3]` | 8 | 66 | {50} | 66 | 8 |
| `tw[0]` | 2 | 0 | {123} | 0 | 2 |
| `tw[1]` | 2 | 0 | {123} | 0 | 2 |
| `tw[2]` | 3 | 50 | {50,23} | 50 | 3 |
| `tw[3]` | 25 | 51 | {50} | 51 | 25 |

were observed for wrappers with more than three test wires. Let us take the wrapper with four test wires as an example. Using Equations 5.1, we derive $p_i = p_o = \left\lfloor \frac{32}{4} \right\rfloor = 8$. The scan chains are presented in Table 5.6 for our approach and for (GOEL; MARINISSEN, 2003a).

The conventional approach (GOEL; MARINISSEN, 2003a) does not require constraints when balancing the test wires. When there is the situation where a single test wire is the bottleneck for the scan time (as the test wires `tw[0]` and `tw[1]` in the example), the previous approach presents slight better test length since their approach can distribute better the wrapper cells, resulting in smaller scan-in and scan-out length. In conclusion, the constraint of equal number of SDI and SDO elements per test wire might increase the scan-in and scan-out length compared to the conventional approach.

On the other hand, our approach might be faster than the conventional approach. It can be observed in Equation 5.5 that $t_i \leq si$, which means that the scan time $t_i$ used in our approach is at most equal to the scan length $s_i$ used in the conventional approach. In conclusion, if both proposed and conventional approaches have the same scan length, the proposed approach might have a smaller scan time, resulting in a faster core test length.

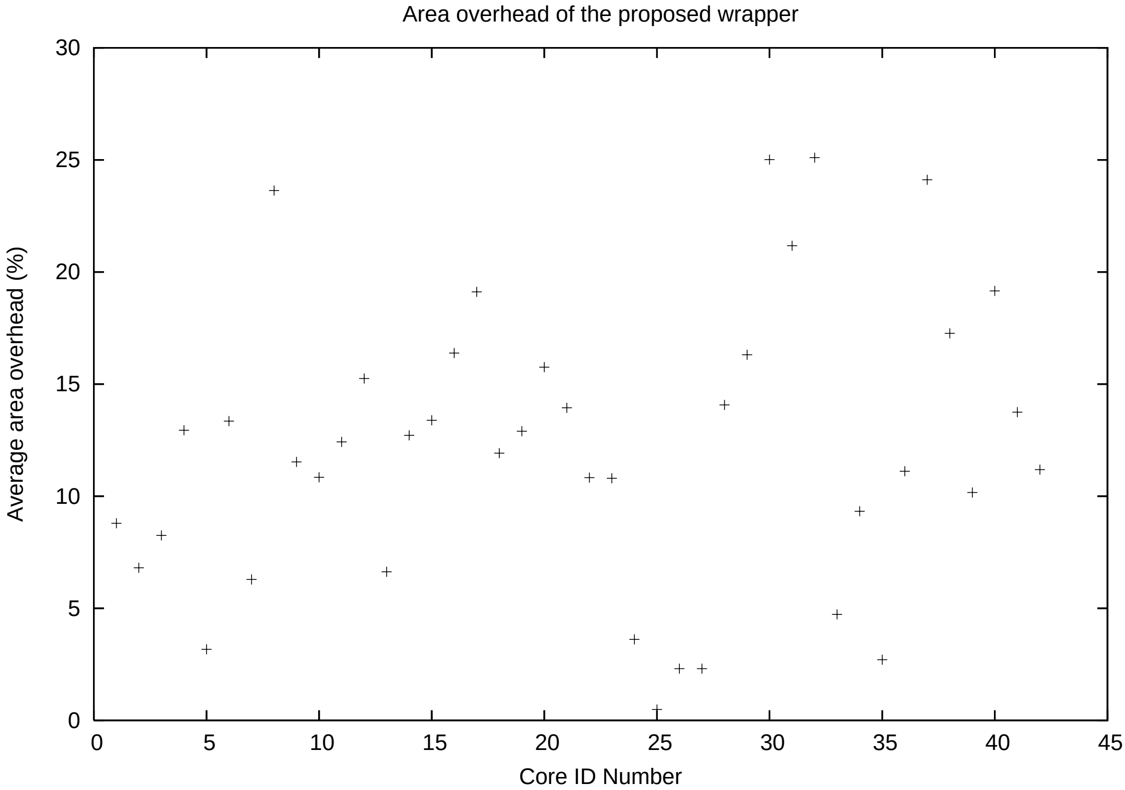
## 5.6 Summary

We present the proposed core test wrapper design which allows test data to be transported via functional protocol ports, such as AXI, DTL, and OCP ports. This approach allows test data transport via an existing functional on-chip bus, NoC, or other functional interconnect. We assume that the test sources and sinks are either an external ATE, or an on-chip BIST engines. We require that the bus, NoC, or other functional interconnect provide guaranteed bandwidth and constant latency, necessary to operate streaming scan tests. We try to exploit the available functional bandwidth as much as possible, in order to minimize the core's test length.

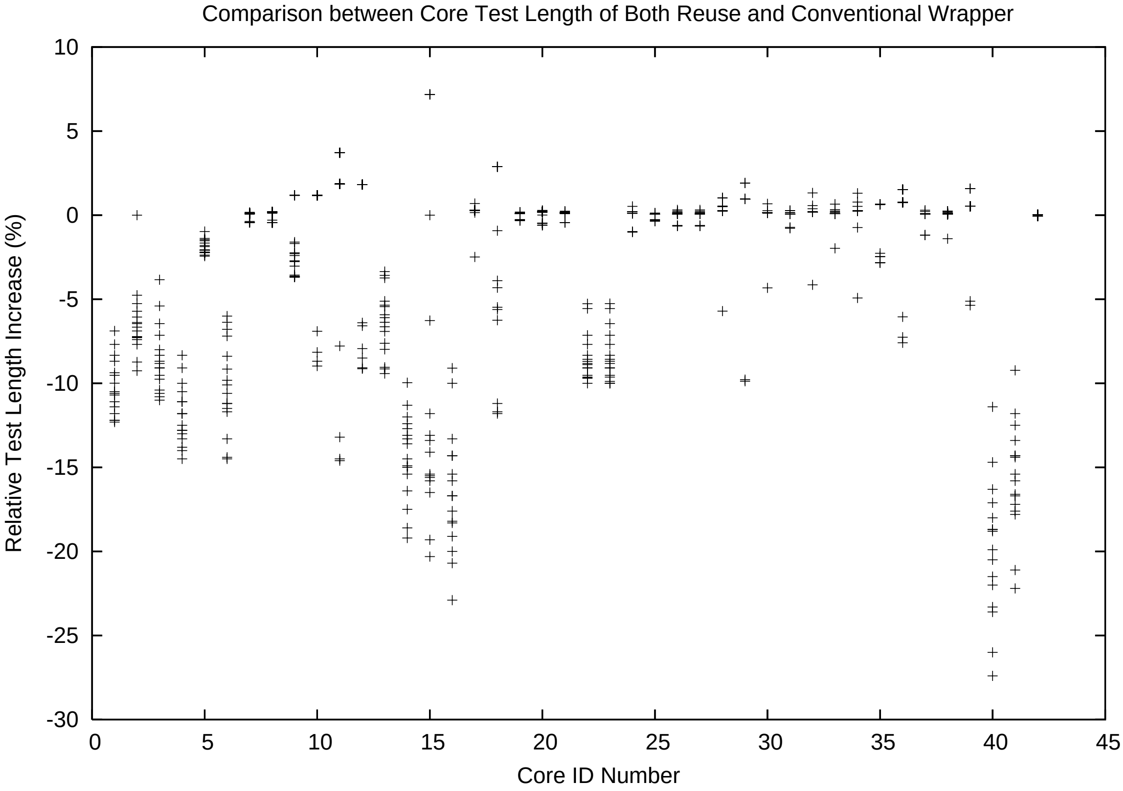
Our new wrapper design no longer relies on a dedicated TAM, and hence has no dedicated TAM ports. Instead, the test data is transported via selected functional protocol ports. This requires careful wrapper design, such that (1) the number of wrapper chains is tuned to the available test data bandwidth; (2) the functional protocols are correctly executed while the test is running; and (3) and the protocol port wrapper cells are positioned at the head and tail of the various wrapper chains.

Although other papers have described the reuse of functional buses and NoCs as TAM, no previous papers have worked out the details of wrapper design for this scenario. The benefits of our approach are that no dedicated TAM is required, that test access to the embedded cores is guaranteed, and that the test (protocol) expansion becomes simpler. By using guaranteed communication services, our approach hides any internal details of the functional interconnect. For the functional interconnect, the transport of test data is just another application, enabling unchanged re-use of the functional interconnect design flow. We can also use the proposed approach on NoCs without guaranteed services by adding a FIFO to the wrapper.

The implementation of the new wrapper design for 672 example cases based on the ITC'02 SoC Test Benchmarks showed an average wrapper area increase of 14.5%, compared to a conventional wrapper. Note that at SoC level, this area increase is negligible. The same 672 example cases show an average reduction of test length of 3.8%. The wrapper silicon area might increase from 0% to 44% when BE NoCs are used, thus, the area overhead for BE NoCs is around 20% .



(a)



(b)

Figure 5.7: Average relative wrapper's gate-equivalent count increase (a) relative core's test length increase of the proposed wrapper with the conventional wrapper design. Results for 42 cores from ITC'02 benchmark and $w \in \{1..16\}$.

# 6 DFT FOR SOURCES AND SINKS

This chapter presents two approaches to implement sources and sinks in NoC-based SoCs: based on the reuse of embedded processors and the ATE interface.

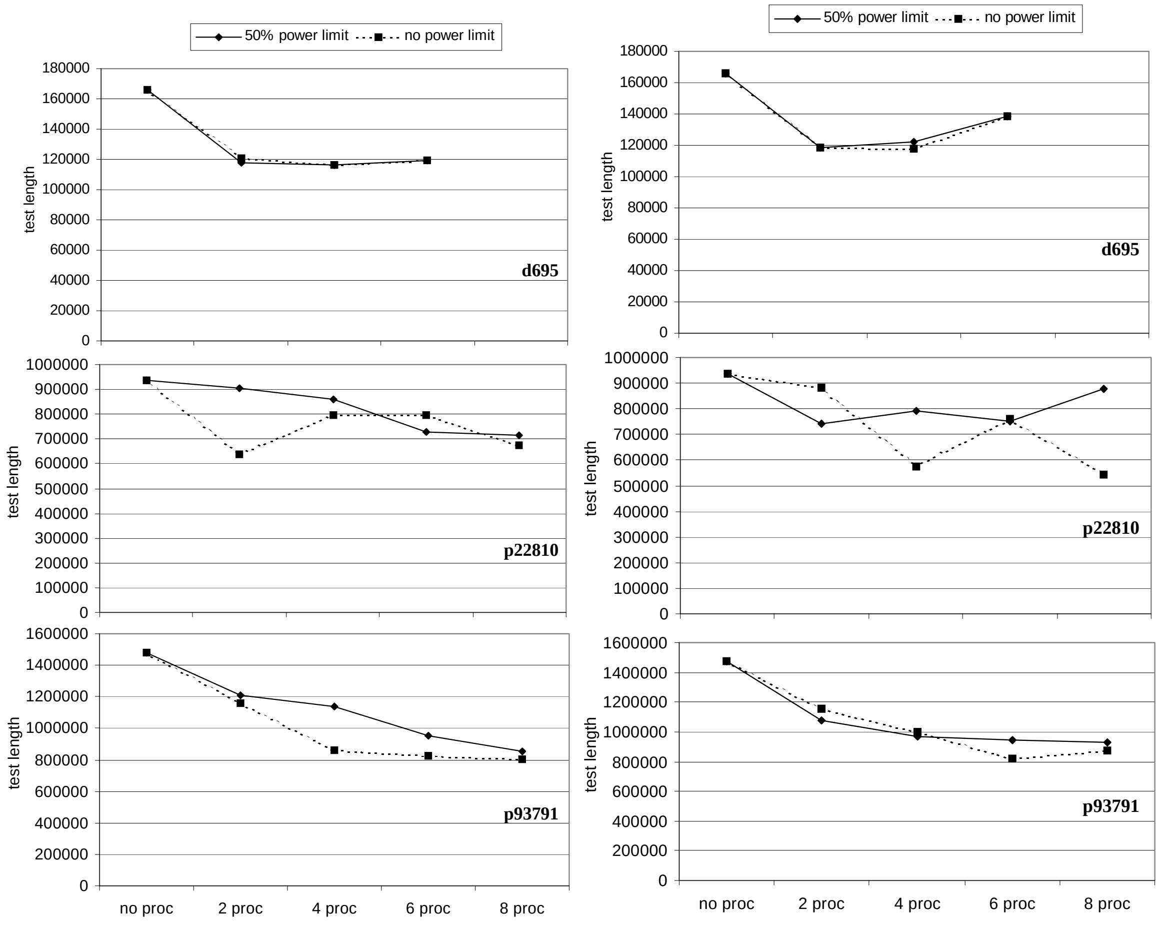
## 6.1 Reuse of Embedded Processors

Current complex systems are using 10 to 15 processors and, according to (HENKEL, 2003), there is an emerging trend pointing to a "sea of processors" with hundreds of heterogeneous processors connected through a NoC. Considering this trend and the increasing complexity for testing such state-of-the-art systems, the reuse of both the processor and the on-chip network for test seems a promising approach to reduce test costs.

*Processor reuse* is based on software-based test of cores, i.e. embedded processors are loaded with test programs to test other cores. Some advantages are that the processor is already connected to the NoC, so there is no need for additional hardware (like the ATE interface), and a cheaper external test equipment can be used since part of the test is embedded into the chip (KRSTIC et al., 2002). However, there are two important drawbacks for the processor reuse during test. In all the previous approaches (AMORY; OLIVEIRA; MORAES, 2003; HUANG et al., 2001a; HWANG; ABRAHAM, 2001, 2003; KRSTIC et al., 2002) *the cores are tested serially because they assume a bus-based architecture*, which leads to longer test lengths. In addition, even if more than one processor is available within the chip, the possibility of reuse and parallelism is limited by the bottleneck of bus-based communication architectures.

*NoC reuse* can replace the test dedicated TAMs for test data transportation. NoCs support parallel transactions, which is important to support test parallelism. However, *if few test pins are available, the NoC is sub utilized and the test length increases.* Combining processor reuse and NoC reuse can solve this problem since part of the test is embedded into the chip and the interconnect support parallelism to use multiple processors as sources and sinks. However, more parallelism can increase the power consumption during test. Although the reuse of the embedded processor avoids the addition of new blocks in the system, *the power consumption of the processor* itself, while running the test program, may be quite high.

Amory et al. (2004; 2005) proposed a power-aware test scheduling tool that combines the reuse of multiple embedded processors and the reuse of the NoC. Thus, the main problems of processor reuse (test serialization and power consumption), and NoC reuse (few test pins and slow tester) are addressed.



(a) (b)

Figure 6.1: Test length with different number of reused processors; (a) Plasma processor; (b) Leon processor.

### 6.1.1 Results

Experimental results are presented for three examples based on benchmarks of the ITC'02 SoC Test Benchmarks (MARINISSEN; IYENGAR; CHAKRABARTY, 2002). The test scheduling tool proposed in (COTA et al., 2003) was modified to support processors as source and sink. Two open processor cores and a NoC are included to the ITC'02 benchmarks to evaluate the impact of the software-based test in a NoC-based system. The first processor is a MIPS-like processor named Plasma (OPENCORES, 2006b). The second one is a VHDL model of a 32-bit processor compliant with the SPARC V8 architectures named Leon (GAISLER, 2006). The NoC is based on the RaSoC model, with mesh topology and different number of routers.

The used test programs are a LFSR with support to reseeding for the test sources and a MISR for the test sinks. Both processors were characterized while running these test programs. The features analyzed during characterization were memory requirement for the test program and test data, test throughput (how fast is the test program), and the power dissipated while running the test program.

Figure 6.1 shows that a significant test length reduction can be achieved when embedded processors are reused to generate/analyze test data.

### 6.1.2 Discussion

Although the reuse of multiple processors has the previously mentioned advantages, there are also some practical problems that must be taken into account for the effective use of the approach:

- *High and constant test throughput*: The test program should generate patterns or analyze responses as fast as possible. The faster the test program is, the shorter is the core test length. Constant test throughput is required to maintain the test streaming. For example, if a test program generates a pattern in 100 clock cycles and the next pattern in 350, there may be a gap in the test transfer, which may require the test of the CUT to stall until data is available. Constant test throughput assures minimal core test length for the assumed bandwidth;

- *The test program should fit in the available memory*: If this constraint is not respected, there are two alternatives, both of them expensive. The first would be to increase the processor memory size, but it may lead to excessive area overhead. The other one is to access other unused memory blocks located in a different NoC tile. The problem is that this additional traffic would need to be modeled by the tool. For these reasons, the test memory requirements should fit to the memory available for the reused processor;

- *Test quality*: the application of the test pattern generated by the test program should lead to a high fault coverage. Thus, only pseudo-random test patterns, as used in (AMORY et al., 2004), may not be enough to test real cores;

- *The combination of the above mentioned challenges* makes it hard to conceive a test program with small size, fast, and high test quality. On one hand, pseudo-random test is small in memory requirement, fast, but the fault coverage can be low even if reseeding is considered. On the other hand, data compression could be used to increase the test quality, but they would increase the memory requirement and the test length.

Due to these problems, we started to investigate alternatives methods for source and sink.

## 6.2 ATE Interface

An *ATE interface* is the DfT module that connects the test pins to the NoC. As presented in the test model, Chapter 4, each partition has at least one ATE interface. The ATE interface is the source and sink of the modules in the partition.

The main tasks of an ATE interface are to do *width conversion* between the number of test pins and the network data width, *protocol conversion* to communicate the ATE with the NoC, and *traffic shaping*, i.e. to correct the traffic deformation caused by load fluctuation.

### 6.2.1 ATE Interface in the Proposed Design Flow

This section demonstrates how the ATE interfaces are defined according to the proposed design flow presented in Section 4.8, page 80.

An ATE interface is defined as a tuple $\{w, d, V\}$, where $w$ is the number of test wires and $d$ is the buffer depth. $V$ is a set of tuples $v = \{nburst, header, nword\}$ with the information required by the ATE interface to test a core, where *nburst* is the number of packets, *word* is the number of words in each packet, and *header* is the packet header content where the most important information is the location of the CUT. There is a $v$ tuple for each module tested by an ATE interface.

According to the proposed flow, the parameters $w$ and $V$ are defined in the test scheduling, while $d$ is defined in the DfT optimization.

### 6.2.2 Functional Description of the ATE Interface

This section presents the functional behavior of the ATE interface.

The *initiator ATE interface* reads data from the test pins and sends it to the CUT via the NoC. For all module in $V$, it sends $b$ packets with the header $h$, where each packet has size $w$. It starts sending the header to the NI with the CUT address, then it sends the test stimuli read from the test pins (`test_pins` variable). This procedure repeats until all packet of all modules in $V$ where sent.

Algorithm 6.1: Initiator ATE Interface

```
for all v ∈ V{
    b := v.nbursts;
    h := v.header;
    do{
        w := v.nwords;
        send(h);
        do{
            data := serial2parallel(test_pins);
            send(data);
        }while(w--);
    }while(b--);
}
```

The *target ATE interface* receives test responses from the NoC and sends it to the test pins. For all module in $V$, it reads an expected number of words $w$ from $b$ packets. It discards the header and reads the packet payload, which represents the test responses. The procedure repeats until all words, of all packet of all modules of $V$ were received.

Algorithm 6.2: Target ATE Interface

```
for all v ∈ V{
    b := v.nbursts;
    do{
        w := v.nwords;
        discard(h);
        do{
            read(data);
            test_pins := parallel2serial(data);
        }while(w--);
    }while(b--);
}
```

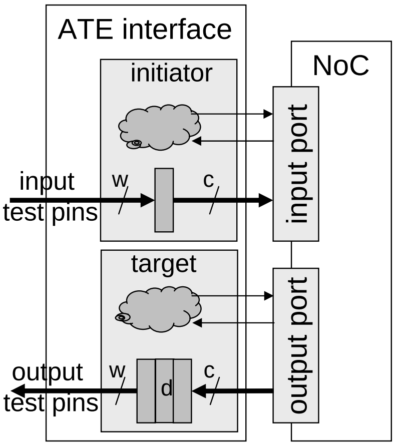


Figure 6.2: Block diagram of the ATE interface.

### 6.2.3 Block Diagram of the ATE Interface

The ATE interface illustrated in Figure 6.2 consists of an initiator ATE interface block and a target ATE interface block. Both blocks do the *width conversion* from $c$ to $w$ wires and vice-versa, implemented by means of a shift-register. The *protocol conversion* must execute the NoC protocol and include packet header information in the test stream, like packet destination and number of words in the packet. The ATE interface also executes the protocol with the NI to enable sending test data while the network is in functional mode. Both NoC protocol and packet format are network dependent, thus, this part of the ATE interface also depends on the target NoC. The *traffic shaping* is carried out by counters that determine the period $d$ and by the FIFO used to eliminate the gaps.

### 6.2.4 Timing Diagram of the ATE Interface

In order to abstract the NoC from the tester, the ATE interface should follow the timing diagram illustrated in Figure 6.3. This figure illustrates the timing of a test stimuli and a test responses path. The ATE sends data every clock cycle in a constant and uninterrupted manner (line 1). One can see that data is sent at constant time intervals of $p_i$ (or $p_o$) clock cycles (line 2) as defined in Equation 5.1, page 86.

Test responses are addressed to the target ATE interface. The received data can be bursty, presenting gaps which would interrupt test data delivery (line 3). The FIFO at the target ATE interface is responsible for recreating the required periodic data (line 4). Finally, data is serialized to the output test pins (line 5).

This proposed timing abstracts the network to a pipe where data flows in a constant and continuous manner, the same way dedicated TAMs do. This pipe has a certain width (i.e. the network bandwidth), determined by the number of pins in the ATE interface, and a certain delay, which depends on the design and length of the pipe (i.e. the number of hops in the test path).

### 6.2.5 Integrating the ATE Interface to the SoC

According to the functional system model, every access to the NoC is via a NI and an on-chip protocol. In some extent, the ATE interface can be seen as a core to

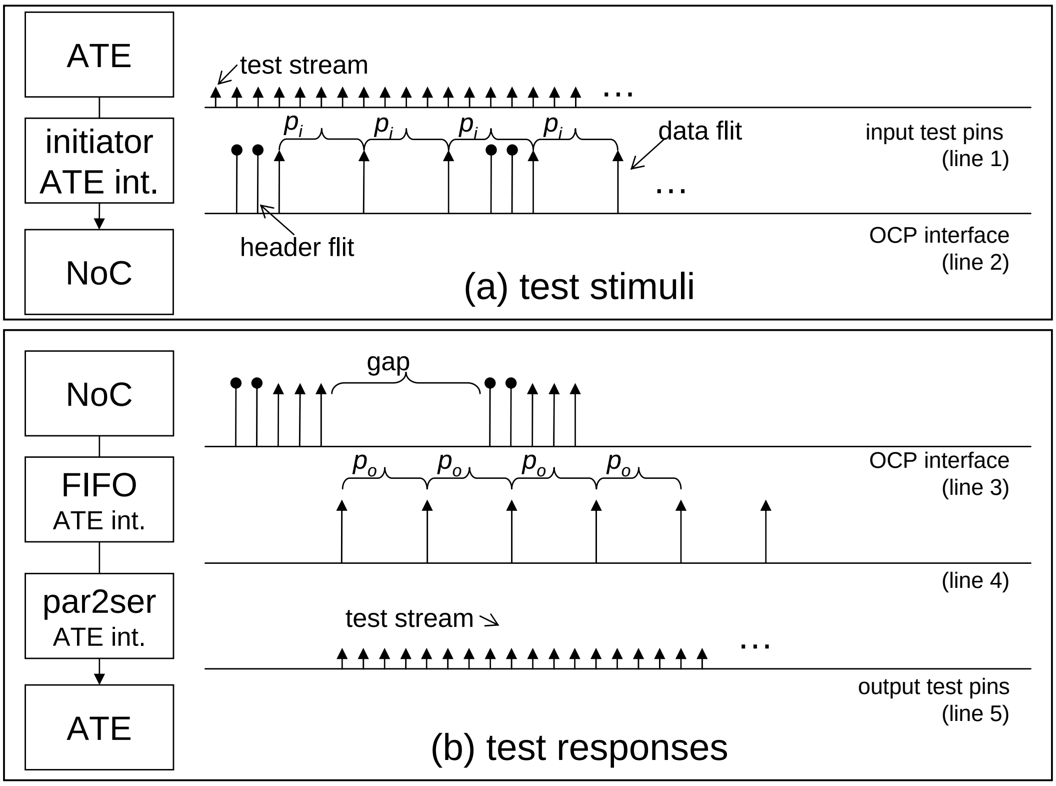


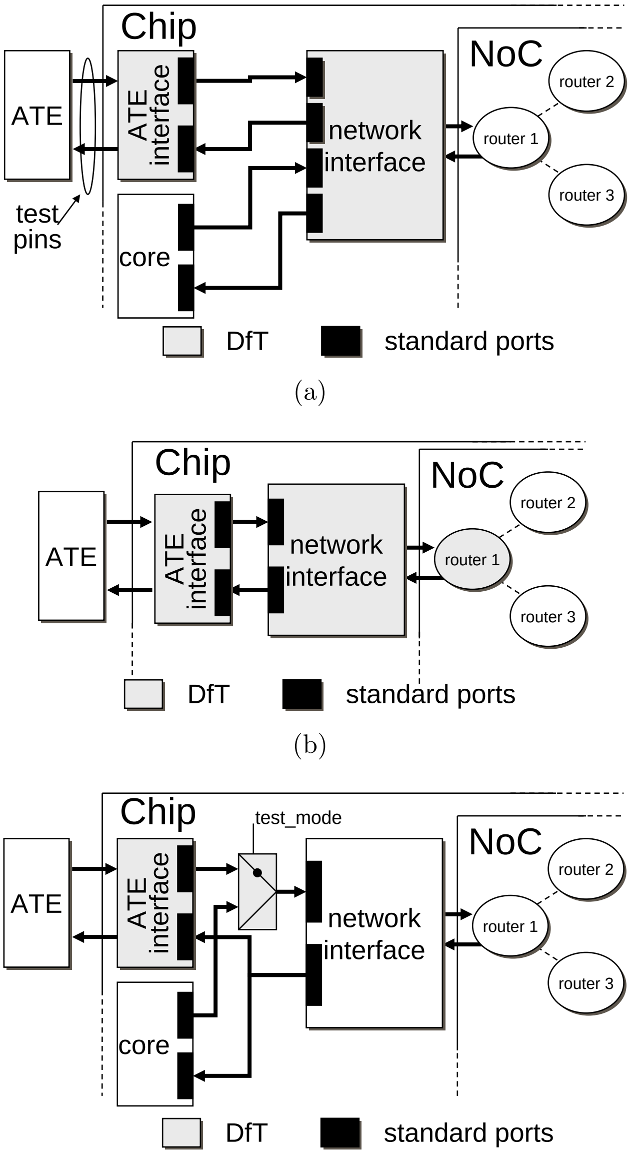
Figure 6.3: Timing diagram of the ATE interface.

the test model, thus, it also requires on-chip ports to be connected to a NI. Hence, the best way to connect the ATE interface to the NI must be found. Let us analyze some design alternatives for an ATE interface.

The first alternative, illustrated in Figure 6.4(a), core ports are added to the NI to connect the test pins. The problem is that adding core ports to the NI requires modifications in the NI kernel. For instance, modifying the number of core ports should be avoided because if there are more cores competing for the router access, the arbitration logic inside the NI changes. Since the NI implementation is not standardized, modifications in the NI would not lead to a general test approach.

The second approach, illustrated in Figure 6.4(b), adds a NI just for test, then, there is no core connected to this NI. This approach requires to find a router without NI or, in case there is no “empty” router, to add a router just to add a NI for test. This alternative modifies the router network and changes the performance characteristics of the NoC, for example, some paths will have additional latency due to a new router. Moreover, the area cost of a router, a NI, and the ports for the test pins are very high.

The third approach, which is the chosen one, is illustrated in Figure 6.4(c). It reuses an existing NI and it multiplexes an existing core port of the NI. The advantages of this approach are the lower area overhead compared to previous alternatives, and it is easier to generalize and automate because it does not require modification in the NI kernel.



(a)

(b)

(c)

Figure 6.4: Alternatives to the ATE interface design; (a) it requires modification in the NI and router; (b) it requires an additional NI; (c) the proposed multiplexed design.

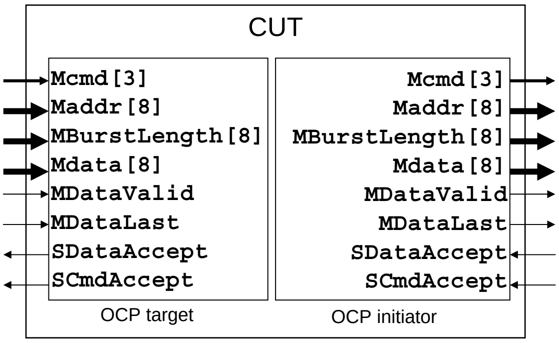


Figure 6.5: Interface of the ATE interface.

### 6.2.6 Synthesis Results

We evaluate the proposed DfT changing network data width $c$ and number of test wires $w$. We show experiments with $c$ equal to 8, 16, 32, and 64 bits. Figure 6.5 shows the OCP interfaces used to interface with the NoC, considering $c = 8$.

The first column of the Table 6.1 shows the network data width. The next three columns show the number of test wires the DfT logic must give support, the minimum packet length, and the minimal FIFO depth. We show these four data because they are the main variables that affect the design, the silicon area of the DfT.

The minimal FIFO depth and packet length required to sustain the test wires are determined by the procedure described in Algorithm 9.1, Chapter 9. The number of FIFO words and the packet length increase because, with more test wires, more bandwidth from the network is required.

The following columns of the table are related to *silicon area to implement the DfT modules.* All modules have been synthesized to $0.35\mu$ technology library. They show the area for the ATE interface (both initiator and target) and the test wrapper for a CUT using the same interface shown in Figure 6.5. The silicon area for the ATE interface initiator suffers a small variance due to different size of some internal counters, like the packet length counter. The ATE interface target is initially smaller than the initiator, but its area increases according to the number of FIFO words. The wrapper area is usually close to the area of an ATE interface since the protocol logic is very similar and the input part of the wrapper may also require a FIFO. The systems with 16/32/64 bits have similar results to the system with 8 bits in terms of silicon area.

These results are presented again in Chapter 9 to discuss the algorithm to find the minimal FIFO size of the DfT logic.

### 6.2.7 Discussion

An ATE interface shapes the test paths. It shapes test paths not only in terms of data width, but also in terms of timing (constant and continuous test data streaming), and in terms of protocol, adapting test and on-chip protocols. For the test point of view, an ATE interface abstracts the NoC such that the tester and the CUTs see the NoC as a dedicated TAM.

The main drawback of the ATE interface design is the possibility of long wires between the test pins and the ATE interface. These long test wires are illustrated

Table 6.1: Area of the ATE interface compared to a wrapper.

| network data width | test wire | packet length | FIFO words | area (eq. gates) | | | |
|---|---|---|---|---|---|---|---|
| | | | | ATE interface (initiator) | ATE interface (target) | wrapper | % (prop/conv) |
| 8 | 1 | 2 | 0 | 713 | 271 | 1023 | 8.4 |
| | 2 | 5 | 2 | 724 | 538 | 1290 | 10.7 |
| | 4 | 14 | 5 | 799 | 836 | 1566 | 13.4 |
| 16 | 1 | 1 | 0 | 968 | 438 | 1558 | 6.8 |
| | 2 | 2 | 1 | 974 | 584 | 1696 | 7.5 |
| | 3 | 4 | 2 | 979 | 926 | 2043 | 9.1 |
| | 4 | 5 | 2 | 976 | 926 | 2033 | 9.1 |
| | 5 | 7 | 3 | 977 | 1080 | 2158 | 9.7 |
| | 8 | 14 | 5 | 982 | 1440 | 2512 | 11.4 |
| 32 | 1 | 1 | 0 | 1476 | 767 | 2593 | 6.2 |
| | 2 | 1 | 0 | 1475 | 752 | 2542 | 6.1 |
| | 3 | 2 | 0 | 1472 | 708 | 2540 | 6.0 |
| | 4 | 2 | 1 | 1487 | 1041 | 2846 | 6.8 |
| | 5 | 3 | 1 | 1473 | 976 | 2858 | 6.8 |
| | 6 | 4 | 2 | 1476 | 1669 | 3523 | 8.5 |
| | 8 | 5 | 2 | 1493 | 1704 | 3513 | 8.6 |
| | 10 | 7 | 3 | 1476 | 1894 | 3726 | 9.0 |
| | 16 | 14 | 5 | 1759 | 2559 | 4362 | 11.1 |
| 64 | 1 | 1 | 0 | 2535 | 1407 | 4615 | 5.8 |
| | 2 | 1 | 0 | 2535 | 1392 | 4561 | 5.7 |
| | 3 | 1 | 0 | 2524 | 1369 | 4548 | 5.7 |
| | 4 | 1 | 0 | 2532 | 1376 | 4510 | 5.7 |
| | 5 | 2 | 0 | 2510 | 1294 | 4509 | 5.6 |
| | 6 | 2 | 0 | 2503 | 1291 | 4509 | 5.6 |
| | 7 | 2 | 1 | 2522 | 2019 | 5148 | 6.5 |
| | 8 | 2 | 1 | 2537 | 1952 | 5141 | 6.5 |
| | 9 | 3 | 1 | 2534 | 1920 | 5144 | 6.5 |
| | 10 | 3 | 1 | 2505 | 1834 | 5153 | 6.4 |
| | 12 | 4 | 2 | 2501 | 3195 | 6491 | 8.2 |
| | 16 | 5 | 2 | 2540 | 3259 | 6481 | 8.3 |
| | 21 | 7 | 3 | 2501 | 3655 | 6893 | 8.8 |
| | 32 | 14 | 5 | 3061 | 4847 | 8092 | 10.8 |

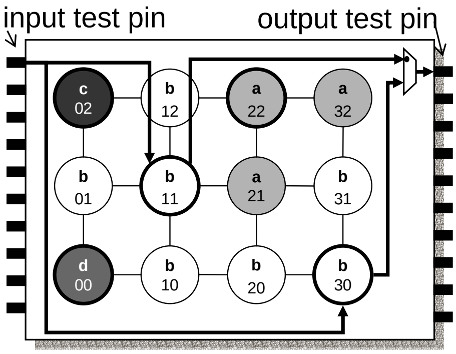


Figure 6.6: Long wires required to connect the test pins to the ATE interface. The fat lines represent the test wires. The circles with fat lines represent the ATE interface location.

in Figure 6.6. The length of the wire depends on the physical distance between the pin and the ATE interface. However, recall that this problem is also present in the dedicated TAM approach. Moreover, the advantage of the proposed approach compared to dedicated TAMs is that it does not need extra wires among the cores.

The second possible drawback is extra mutiplexers required to connect the ATE to the NI. It can increase the delay of the system. Moreover, since the proposed approach supports multiple ATE interfaces per vTAM, the wires assigned to a vTAM must be shared among the ATE interfaces. This is described in Figure 6.6 partition *b*, where the output test pin is shared by two test wires. The solution to support multiple ATE interfaces per vTAM has been chosen because, otherwise, the NoC partition would have to have a shape such that the routing algorithm could reach all nodes within the partition with a single ATE interface. It would impose more severe restriction to the test length. For this reason we assume multiple ATE interfaces per partition. Recall that although there might be multiple ATE interfaces per partition, only one is activated per time to avoid packet collision.

## 6.3 Summary

This chapter presented two approaches to transport test data over a NoC. The first one is based on the reuse of embedded processors as test sources and sinks. The second one presented a module, called ATE interface, required to connect the test pins to the NoC and to shape the path from the tester to the CUT according to the test requirements, abstracting the NoC from the test domain. The results demonstrated that the proposed design for ATE interface is simple, configurable, and the required silicon area is equivalent to a test wrapper. The drawback of the ATE interface is that long wires might be required to connect the test pin to the ATE interface and it may increase the system delay. However, it does not need extra wires among the cores as dedicated TAMs.

The approach that has been continued is the one based on ATE interface because it is based on external testers, which is the most used test application approach, and it is simpler to implement than the processor reuse approach. As presented before, although the reuse approach is cheap in area and the connection to the NoC is available, developing cost-effective test program is the main challenge. Further developments on multiple processor reuse are future work.

# 7 DFT FOR NETWORKS-ON-CHIP

Network-on-Chip has recently emerged as an alternative communication architecture for complex system chip and different aspects regarding NoC design have been studied in the literature. However, the test of the NoC itself for manufacturing faults has been marginally tackled. This chapter proposes a scalable test strategy for the routers in a NoC, based on partial scan and on an IEEE Std. 1500-compliant test wrapper. The proposed test strategy takes advantage of the regular design of the NoC to reduce both test area overhead and test length. Experimental results show that a good tradeoff of area overhead, fault coverage, test data volume, and test length is achieved by the proposed technique. Furthermore, the method can be applied for large NoC sizes and it does not depend on the network routing and control algorithms, which makes the method suitable to test a large class of network models.

## 7.1 Introduction

Several authors have presented different aspects regarding the design and implementation of on-chip networks (GUERRIER; GREINER, 2000; MORAES et al., 2004; BJERREGAARD; MAHADEVAN, 2006). Recently, industrial NoCs have also been proposed (VERMEULEN et al., 2003). Furthermore, the reuse of the NoC as TAM has been presented as a cost-effective strategy for the test of embedded IP cores, with reduced area, pin count, and test length costs (AMORY et al., 2004; COTA; LIU, 2006). Although one may claim that the network operation is also tested when it is transmitting test data, for diagnosis purposes and complete fault coverage it is important to define a test scheme for the network before its reuse as TAM. For this reason, most NoC reuse test strategies assume that it has been tested previously.

Some test approaches for NoCs have been discussed in the literature (AKTOUF, 2002; UBAR; RAIK, 2003; VERMEULEN et al., 2003). Aktouf (2002) suggests the use of a boundary scan wrapper. Other approaches (UBAR; RAIK, 2003; VERMEULEN et al., 2003) suggest that a wide variety of standard DfT solutions can be used, from BIST for FIFOs, to functional testing of wrapped routers. However, those proposals have not been applied, to the best knowledge of the authors, to actual NoCs.

In this chapter, we firstly verify the efficiency of some of the previously suggested NoC test approaches, as well as the applicability of standard DfT techniques (ARABI, 2002; MARINISSEN et al., 2002; WU; MACDONALD, 2003) to NoC testing. Experiments show that existing approaches may lead to considerable area overhead and test length, making the NoC testing a major bottleneck for the system design. Hence, we propose a scalable and cost-effective DfT strategy for the routers of the NoC.

The proposed method is based on partial scan and on an IEEE Std. 1500-compliant test wrapper, and it takes advantage of the NoC regularity. Moreover, the test strategy is scalable and independent of the network functional operation, which makes it suitable for a large class of network models and implementations. The method is applied to three versions of a NoC model with different sizes to demonstrate its effectiveness. The results are analyzed in terms of area overhead, test length, test data volume, fault coverage, and power dissipation.

The contributions of this proposal are twofold: firstly, it shows that the application of standard DfT techniques to the NoC testing is not straightforward, and may lead to excessive costs if applied deliberately. Secondly, it presents a structured, scalable, and cost-effective test scheme for NoC routers.

This chapter is organized as follows: Section 7.2 presents a brief overview of NoC design and a description of the network used in the sequel of the chapter. Section 7.3 presents the main challenges for testing NoCs. Section 7.4 presents the results of the application of standard DfT methods in the test of a NoC. Section 7.5 describes the proposed test strategy, while Section 7.6 discusses the experimental results. Sections 7.7 and 7.8 discuss limitation of the approach and present the conclusion.

## 7.2 Case Study: the SoCIN Network

In order to evaluate the proposed test strategy, a packet-switched network named SoCIN (System-on-Chip Inter-connection Network) (ZEFERINO; SUSIN, 2003) is used. The router that implements the network protocol is called RASoC (Router Architecture for System-on-Chip). This router uses input buffering, a round-robin algorithm for arbitration, a handshake algorithm for flow control, and an oblivious routing algorithm. Switching is based on the wormhole approach, where a packet is broken up into flits (flow control units). The channel width used in the experiments is 20 bit-wide (16 data bits and 4 control bits). RASoC has four ports to connect to its neighbors and one port to connect to the embedded IP core. The network implements a 2D torus topology, where each router is configured with 16-bit flit width and FIFOs depth equal to 4.

## 7.3 Main Challenges

Table 7.1 presents some characteristics of the original RASoC router, without test circuitry. The characteristics of a processor core called Plasma (OPENCORES, 2006b) are also presented in the table for comparison.

Plasma is a typical example of small-to-medium size IP core. The processor is compatible to MIPSI instruction set, and has a 32-bit multiplier, a shifter, a 32x32 register bank, and 3-stage pipeline. Plasma has less IO pins, and consequently, a smaller area for the test wrapper than the RASoC router. In addition, in spite of the 32x32 register bank, Plasma has a lower density of flip-flops (36%) than RASoC (45%).

The power consumption presented in Table 7.1 is characterized considering the

Table 7.1: Comparing the RASoC router and the Plasma processor.

| | # gates | # flip-flops | flip-flop relative area | IO pins | power ($\mu w$) |
|---|---|---|---|---|---|
| RaSoC | 4605 | 425 | 45% | 202 | 2.24 |
| Plasma | 20118 | 1444 | 36% | 105 | 6.12 |

dynamic and static power consumption of the module. To evaluate the dynamic consumption, the module is initially synthesized to an ASIC technology library for which the power consumption of technology cells is available. The resulting netlist description is simulated. During this simulation, the switching activity of each cell of the technology library is captured. Then, the power consumption per clock cycle is computed by multiplying the number of toggles and the power consumption per cell. The total power consumption for the whole simulation is given by the average power consumption of the module in each clock cycle. Plasma power consumption was characterized for an arbitrary application. For RASoC, the power was characterized considering four packets being routed in parallel, which maximizes the switching activity. All packets have the same number of flits and random payload data. One can observe that Plasma presents higher power consumption than RASoC, since Plasma has more flip-flops, which are the main source of power consumption. However, considering the power per gate measure, RASoC has a ratio (2.24/4605) of 0.5nw/gate against (6.12/20118) 0.3nw/gate of the Plasma processor. Thus, although a single router is smaller than most functional IP cores, it may have a higher switching activity. Therefore, the total power consumed by the NoC may easily be higher than the consumption of other IP cores in the system.

This comparison between RASoC and Plasma highlights the challenge to find a cost-effective test strategy for NoCs. The router has fewer gates per I/O port, higher density of flip-flops, and higher power consumption per gate. Those features indicate, for instance, that test wrapper and full-scan implementations may be too costly if applied to each router independently. On the other hand, NoCs usually have very regular designs. Although different implementations of routers exist, most of them follow the conceptual model presented in Figure 2.6, page 39. The combination of the regularity of a NoC and the predictable router structure is explored in this work to reduce test costs.

## 7.4 Evaluating Standard Test Strategies in NoCs

Some authors argue that the NoC is another IP core (flat or hierarchical) in the system. In this case, its test can be defined using traditional core-based testing strategies (UBAR; RAIK, 2003; VERMEULEN et al., 2003). This means the use of an IEEE Std. 1500-compliant test wrapper (MARINISSEN et al., 2002) and the use of scan-based approaches to test the routers. Considering the NoC as a *flat core*, a single test wrapper is inserted in the NoC interface. Otherwise, if the NoC is considered as a *hierarchical core*, one test wrapper for each router is necessary.

Considering the regular design and the presence of identical IP cores in the NoC structure (routers), test strategies previously proposed for similar systems may

Table 7.2: Standard test strategies applied to a 3x3 NoC.

| | flat core full scan | boundary scan | hier. core - full scan | hier. core - full scan with comparator |
|---|---|---|---|---|
| Total area (eq. gates) | 49437 | 79290 | 62037 | 62994 |
| area overhead | 19% | 91% | 50% | 52% |

be applied to the on-chip network. Wu and MacDonald (2003), and Arabi (2002) propose test strategies for *identical IP cores* in a single chip. Both approaches require full-scan and IEEE Std. 1500 wrapped cores with registered I/O pins. The test consists on applying test patterns to all identical IP cores in parallel and comparing the responses within the chip. Whenever a fault is detected, a special circuit at the output of each IP core allows to individually test each block for diagnosis. Those approaches result in a considerable reduction in test length (due to the test parallelism), in test volume (do not require storage of the test response in the tester), and in ATPG CPU time (ATPG runs for a single IP core).

Another approach, proposed by Aktouf (AKTOUF, 2002), is a *boundary-scan based strategy* for testing massively parallel machines. The method takes advantage of the regularity of the communication architecture to reduce the test length. Boundary scan cells involve each router in the architecture. Full scan routers are assumed.

We evaluate these four test configurations with respect to the area overhead and results are presented in Table 7.2. The reference design is a 3x3 SoCIN network with nine RASoC routers implementing a torus topology as described in Section 7.2. The size of this NoC without test circuitry is 41,445 gates. The second column in Table 7.2 (*Flat core full scan*) shows the results when the network is considered as a single IP core, i.e., a single test wrapper and a full-scan strategy are implemented for the network as a whole. Column 3 (*Boundary scan*) shows the results for the boundary-scan approach. In this configuration, all routers have full-scan and a boundary scan test wrapper, as proposed in (AKTOUF, 2002). The last two configurations assume the network is a hierarchical IP core, i.e., each router is treated independently. In the third configuration, shown in Column 4 of Table 7.2 (*Hierarchical core-Full scan*) each router has a test wrapper and implements a full scan testing. The fourth test model (column *Hierarchical core-Full scan with comparator*) repeats the third one, but includes internal comparators, as proposed in (ARABI, 2002; WU; MACDONALD, 2003), to reduce the test volume. The fault coverage for all configurations is above 98%.

One can observe in Table 7.2 that the area overhead for three out of four approaches is prohibitive. The only exception is the first configuration, which considers the NoC as a flat core with full scan. Although this configuration presents an affordable area overhead, the flat approach typically ignores the internal organization of the IP core. Indeed, we show further in this chapter that less area overhead can be achieved when the internal structure of the NoC is considered. Therefore, new methods must be developed to meet the specificities of this new communication platform. Such a method is proposed in the next section and applied to the same 3x3 SoCIN network for comparison.

## 7.5 Proposed Test Strategy for NoCs Based on Identical Routers

We propose a NoC testing approach that combines the best features of the "*Flat core full scan*" approach (reduced area overhead) and the "*Hierarchical core Full scan with comparator*" (reduced test volume) configurations explained in Section 7.4. The proposed strategy considers the NoC as a flat core (a single test wrapper for the whole network is required) but it does not require a full scan implementation, which further reduces the area overhead. Moreover, different from the flatten approach, we explore the regular design of the NoC to reduce test length and data volume. The proposed strategy is presented in three parts:

a) The router testing, which shows how to configure the router internal scan chains to reduce area,

b) The NoC testing, which explains how the scan chains of the routers are connected together at NoC level, and

c) The NoC test wrapper, which details the definition of the IEEE Std. 1500 compliant test wrapper for the NoC exploring the network regular structure to reduce area.

### 7.5.1 Router Testing

A router is composed by control logic (routing, arbitration, and flow control modules) and input FIFOs. Control logic is considerably simpler to be tested, because it contains a small number of flip-flops and gates. The input FIFO poses the main problems for the router testability. Figure 7.1(a) illustrates the architecture of the primary inputs of a router, i.e., the FIFO implementation. If full scan is directly implemented in this structure, all flip-flops of the FIFO will become scan flip-flops and will be chained together. In the proposed approach, we split the FIFO and define a single scan chain using only the first position of the queue, as illustrated in Figure 7.1(b). This single scan chain provides the controllability and observability required to test the whole structure, since the FIFO is usually not very deep and there is no feedback logic in this block. Thus, any sequential ATPG tool can generate test patterns for the whole FIFO with an affordable effort.

To complete the router testing, a second scan chain is defined with the remaining flip-flops of the control logic, which are the flip-flops used to implement the routing algorithm, for example. This approach avoids expensive solutions like full-scan and BIST (UBAR; RAIK, 2003; VERMEULEN et al., 2003), while the ATPG tool can still generate high fault coverage at reasonable CPU time.

### 7.5.2 NoC Testing

After defining the test structure for each router, one must define an access mechanism to transmit test data from the network interface to each router and vice-versa. We propose a general test communication protocol, which can be applied to regular

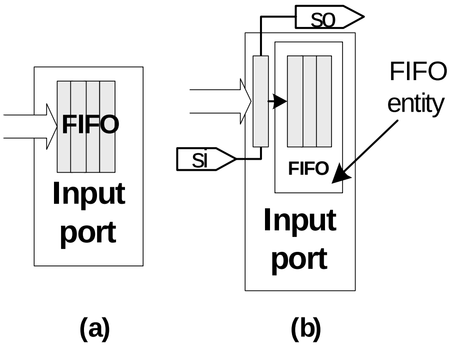


Figure 7.1: Splitting the input FIFOs: (a) original and (b) modified for testing.

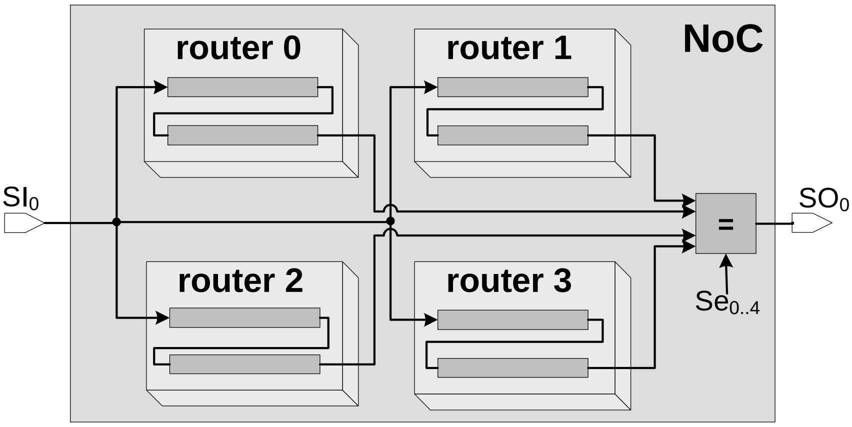


Figure 7.2: Testing multiple identical routers.

NoC topologies, such as mesh and torus. In this protocol, test patterns, coming from the external tester, are simultaneously applied to all identical routers. Test responses of the routers, on the other hand, are internally compared, as proposed in (ARABI, 2002; WU; MACDONALD, 2003). If test responses are different, a mechanism for diagnosis can be activated. Otherwise, the test continues.

Test vectors are broadcasted to routers by a single pin in the network interface, as shown in Figure 7.2. Note that only test related ports and wires are shown in Figure 7.2. The block denoted by the equal signal indicates a comparator that checks test responses against each other. The number of comparators required in the NoC depends on the number of scan chains in the routers. For each scan chain in the routers there must be a comparator. In Figure 7.2, for instance, a single scan chain per router is assumed, and the four chains feed the single comparator. Ideally, all routers are tested in parallel and a single comparator is used. However, there may be limitations in the maximum fanout of the scan input pin (SI in Figure 7.2) and in the test length achieved by a single scan chain in the routers. The NoC designer can define, then, an alternative solution by increasing the number of scan chains per router (area overhead does not change) to reduce test length and increasing the number of comparators, whose area can be easily estimated.

#### *7.5.2.1 The Comparator*

Figure 7.3 presents a circuit for output comparison, similar to the modules proposed by Wu and MacDonald (2003), and Arabi (2002). When running in test

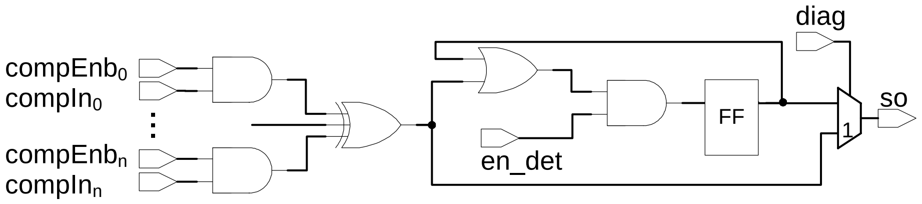


Figure 7.3: Comparator block.

mode, ports `compEnable` (`compEnb`$_x$) and enable detection (`en_det`) are assigned to '1', while the diagnostic port diag is set to '0'. Signals `compInput` (`compIn`$_x$) receive one scan chain output of each router. All corresponding bits unloaded from each router scan chain are compared against each other, as depicted in Figure 7.2. If there is any difference, the xor gate generates an error signal '1' in the `so` pin.

The comparison logic also supports diagnosis. In diagnosis mode, signal `diag` is initially set to '1'. Then, signal `compEnb`$_x$ corresponding to a single router is set to '1', while signals `compEnb`$_x$ of the remaining routers are set to '0'. Test vectors are applied again to all routers, but the output of only one router will be captured. This procedure is repeated until the defective router is found. Notice that other testing approaches that consider the NoC as a non-hierarchical core are not able to identify the router with defect since they abstract the internal structure of the network. On the other hand, the hierarchical approach has a large area overhead. However, our approach is based on the flat design but structured in such way that the defective router can be identified.

### 7.5.3 Test Wrapper for NoCs

To complete the definition of the test strategy, we briefly describe an IEEE Std. 1500-compliant wrapper implementation for the network.

In order to support on-chip test response comparison, one must provide the same test stimuli for all the routers. Thus, functional pins and scan chains of the routers must receive the same test stimuli. Identical stimuli for scan chains are provided by the strategy presented in Section 7.5.2. However, since there is a single test wrapper for the whole NoC, the test wrapper design must also ensure that the functional ports of the routers receive the same test stimuli.

Figure 7.4 presents the proposed IEEE Std. 1500-compliant test wrapper for the NoC. In test mode, functional inputs receive test patterns through the $ci_x$ cells, as in the original test wrapper. However, the difference from the original IEEE Std. 1500 test wrapper design is that the number of $ci$ cells is not equal to the total number of functional inputs of the NoC. Instead, there are as many $ci$ cells as the network channel bitwidth, i.e., the number of input pins necessary to connect the NoC to one IP core. In Figure 7.4, we are assuming 20 bits (see $ci_0$ to $ci_{19}$). Within the wrapper, these input pins feed the functional inputs of each router.

Similarly, the number of wrapper test output cells is smaller than the number of network functional outputs. Each functional output pin of the NoC is connected to a comparator, similar to what is done with the routers scan chains. Comparators results are chained together ($co_1$ to $co_{19}$ in Figure 7.4) and assigned to a single wrapper scan output pin. For the wrapper shown in Figure 7.4, for instance, there are 20 comparators, since we are assuming routers outputs of 20 bits (bitwidth of

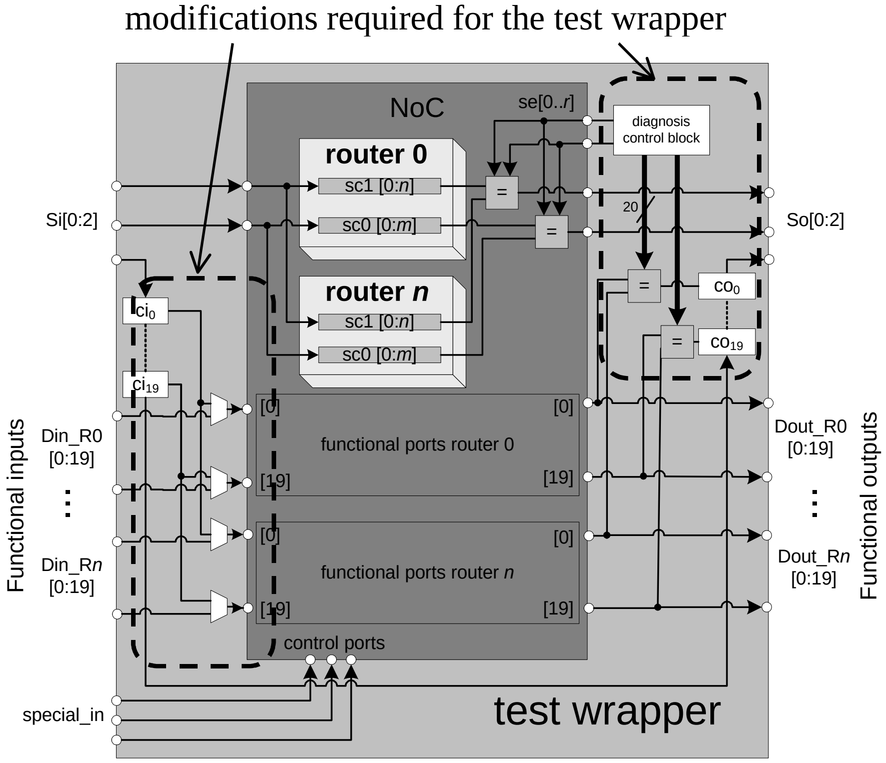


Figure 7.4: Proposed test wrapper for NoCs.

the channel connecting the NoC to the IP core). Such a structure reduces not only the area overhead of the wrapper (by reducing the number of wrapper scan cells), but also the NoC test length (by reducing the number of shift operations during test). The wrapper cell definitions are compliant with the IEEE Std. 1500.

It can be observed in Figure 7.4 that, for example, functional inputs `Din_R0[0]` to `Din_Rn[0]` receive the same value through *ci* wrapper cells during test and the comparator presented in Figure 7.3 is attached to the functional outputs. The result from the comparison can be loaded in the *co* wrapper cells for scan out. During the diagnosis mode, the diagnosis control block is responsible to set the `diag` and $\texttt{se}_x$ ports, presented in Figure 7.3, to the appropriate router.

We note that the test access mechanism that connects the network to the system interface and the external tester is defined by the SoC designer, since the network is considered as another IP core in the system.

## 7.6 Experimental Results

We evaluated the proposed test strategy for three different network sizes, to show the scalability of the method.

All the experiments were carried out using DFTAdvisor (scan insertion tool) and Fastscan (ATPG tool) (MENTOR GRAPHICS, 2007) from Mentor Graphics, using the ADK (TSMC 0.35) technology library. For the evaluation of the fault coverage, stuck-at model is assumed and all faults classified by the tools as possibly detectable are considered undetected faults. The ATPG was executed on a Pentium 4 2.6GHz with 1G RAM running Linux OS.

Table 7.3 presents the testing results using the proposed approach for each net-

Table 7.3: Results for the proposed test strategy.

| **NoC size** | **Area costs** | | **Test efficiency** | | | | **Test time (cycles)** | | | **CPU time (s)** |
|---|---|---|---|---|---|---|---|---|---|---|
| | Original area (#gates) | Area with proposed DfT (#gates) | fault coverage (%) | # patterns | test volume (bits) | # collapsed faults | 3 unbalanced chains | 3 balanced chains | 8 balanced chains | |
| 3x3 | 41445 | 45107<br>(+8.8%) | 98.82 | 383 | 254465 | 105024 | 32187 | 21407 | 9087 | 2876 |
| 4x4 | 73680 | 79923<br>(+8.4%) | 98.93 | 395 | 273845 | 186702 | 33271 | 22155 | 9451 | 11350 |
| 5x5 | 115125 | 124616<br>(+8.2%) | 98.93 | 466 | 315065 | 291712 | 39286 | 26182 | 11206 | 33916 |

work size. Columns 2 and 3 show, respectively, the area of the original NoC and of the NoC with the proposed test structures. The area is measured in number of gates and does not include wiring length. Column 3 also presents the percentage of the increase in the area due to the DfT hardware. The fault coverage, the number of test patterns, the test volume, and the number of collapsed faults are presented in Column 4 through 7, respectively. Different test lengths are presented for each network in Columns 8, 9, and 10, to demonstrate that the designer can use different scan configurations for the routers. Finally, the CPU time spent by the ATPG is shown in Column 11.

The proposed approach has three sources of area overhead: the partial scan chain in the router, the comparators in the NoC, and the test wrapper of the NoC.

Although not shown in Table 7.3, the area of the router with the proposed partial scan is 4,859 gates, which represents an overhead of 5.5% over the original router. A full-scan implementation in this structure results in 14.7% of area overhead. Notice in Table 7.3 that the area overhead of the proposed method for the 3x3 NoC (+8.8%) is smaller than the one achieved by the pure application of full-scan, presented earlier in Table 7.2 (19%).

For the sake of a fair comparison of wrapper approaches, let us assume a configuration where the proposed partial scan approach for the routers is combined with the traditional IEEE Std. 1500 wrapper. In this case, the area of the resulting NoC is still smaller (46,756 gates compared to the 49,437 gates presented in Table 7.2). This same configuration (traditional test wrapper) implemented in the 5x5 NoC results in 14.2% of area overhead against 8.2% if the optimized wrapper is used.

It is important to notice that the area overhead of the router test structures decreases as the FIFO depth increases (scan chains are inserted only in the first FIFOs word). Hence, using the proposed approach, more complex network implementations will present a better tradeoff of test costs. Moreover, the area of the test wrapper increases linearly with the number of routers (one wrapper scan pin per router scan cell), and, most importantly, it increases sub linearly with the channel width, since the number of wrapper scan cells for the routers functional inputs may be kept constant.

We did not find in the literature the size of a complete system using NoCs. Therefore, we cannot estimate accurately the overall increase in the chip size. Nevertheless, according to the size of the systems evaluated in (RAJSKI et al., 2003), the current system sizes range from 0.5 to 10 million gates. We estimate, however, the overall contribution of the NoC testing circuitry to be about 0.7-1.9% for a 0.5 million-gate design, and 0.04-0.095% considering a 10 millions-gate design.

The test data volume presented in Table 7.3 does not consider the expected responses, since they are evaluated on-chip. Considering the same test patterns and their corresponding test responses, the total test volume would be 382,004, 405,380, and 470,243 bits, for the 3x3, 4x4, and 5x5 NoCs, respectively. Thus, the test volume saving of the circuit presented in Figure 7.2 is of 32%. Nevertheless, one can observe that high fault coverage is still achieved independent of the number of routers in the NoC and for the same relative cost in area.

The NoC designer can define different scan chains configurations to connect the routers functional inputs in the wrapper, thus generating different test lengths. Some possible configurations are presented in Table 7.3 (Columns 8, 9, and 10). Considering the 5x5 NoC with eight balanced scans, for instance, the test length using a standard test wrapper would be 80,002 clock cycles, while the proposed wrapper achieves 11,206 clock cycles.

Finally, power consumption may be an important limitation to the test parallelization within the NoC. Hence, we have characterized the power consumption per router during test. The power consumption is characterized considering the dynamic and static power consumption during the test execution and the actual test patterns. To evaluate the dynamic consumption, the router is initially synthesized to an ASIC technology library for which the power consumption of technology cells is available. The resulting netlist description is simulated with the actual test patterns calculated by the ATPG tool. During this simulation, the switching activity of each cell of the technology library is captured. Then, the power consumption per clock cycle is computed by multiplying the number of toggles and the power consumption per cell. The total power consumption for the whole simulation is given by the average power consumption of a router in each clock cycle of the test procedure. For the network configurations presented in Table 7.3, the power consumption per router is $4.34w$. Since all routers receive exactly the same test patterns, the power of the NoC depends only on the number of routers. Thus, considering a 10 million gate design (RAJSKI et al., 2003), the total area of a NoC is small (less than 1% considering 3x3 NoC), and then the contribution to the system power consumption is expected to be small too. However, if the test power of the NoC becomes an issue, one can test groups of routers at a time instead of all of them in parallel.

Routing the comparators may cause routing congestion, timing and power problems because of the long wire length. In these situations, one can use comparators between neighbor routers, thus reducing the wiring length. We are considering manufacturing faults, which are usually randomly distributed in the design. Thus, the probability of multiple errors that produce exactly the same output is low.

Figure 7.5 shows the scalability of the proposed method. As the size of the NoC increases (see number of gates curve) the test data volume and test length increase in a much lower rate, while the fault coverage is kept constant and the area overhead is reduced. This graph demonstrates that the approach can scale to test very large NoCs that support connecting several IP cores.

## 7.7 Limitations

The main limitation of the proposed approach is that it is applied only to NoCs with large number of identical routers. Other approach must be proposed for NoCs based on non-identical routers.

The second limitation of the proposed approach is that it does not test the NoC

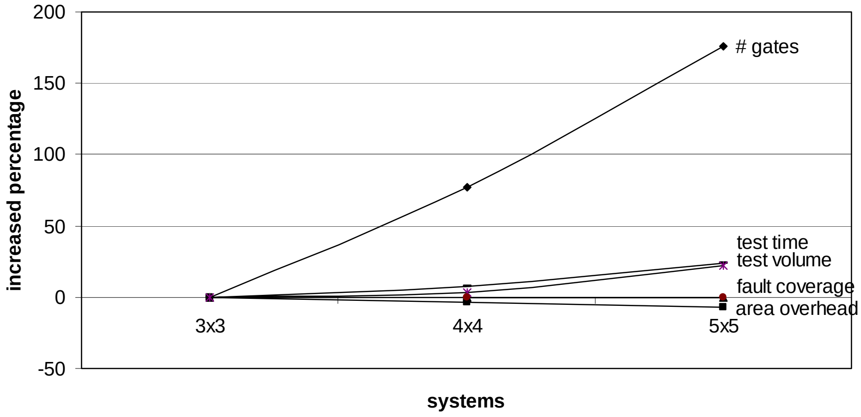


Figure 7.5: Test costs versus NoC silicon area.

channels. However, a complementary approch for NoC channel testing has been accepted for publication (KASTENSMIDT et al., 2007).

The main limitation of the experimental results is that it does not evaluate the wiring area to implement the comparison modules. Since the routers of a NoC are spread in the chip, the wires from the routers to the comparator block might be long.

The proposed test model suggests that routers and cores might be tested in parallel, where each router is considered a core and is tested individually. However, the proposed approach tests the entire NoC at a time. In conclusion, the proposed test approach is not totally compatible with the overall test approach for two reasons: it has limited scope and a router cannot be individually tested.

Compatible test solutions for testing NoCs were investigated, however, they lead to a high area overhead (about 30%) because when a router is considered a core, it requries a test wrapper, which increases the area overhead. Approaches based on fewer and less expensive wrappers for routers must be proposed.

## 7.8 Summary

We have shown that routers of on-chip networks pose additional challenges to find a cost-effective test strategy when compared to functional cores. The large number of I/O pins, small area, and high density of flip-flops make the application of standard DfT techniques a more complex task for those structures. However, the test cost of a NoC can be significantly reduced if its regular design is considered.

We have proposed a test strategy for NoCs based on partial scan and an IEEE Std. 1500-compliant test wrapper, which reduces the test length and area overhead by exploiting the NoC regular design. An academic network has been used to demonstrate the feasibility of our approach. Observing the obtained results one can conclude that the proposed test strategy is indeed a cost-effective solution for the test of routers that compose an on-chip network. We reduced the ability to isolate the routers for test for the sake of area overhead reduction, while keeping a high fault coverage, low test length, and low test data volume. The proposed test strategy can be implemented using only a scan insertion tool and an ATPG tool.

The main limitations of the proposed approach are that: it cannot be used to test NoCs without large number of identical routers, it does not test NoC channels, the results do not evaluate the area overhead to implement the wiring from the router to the comparison block, and it is not totally compatible with the overall test approach. Other approaches must be found to fill these gaps.

# 8 TEST SCHEDULING

This chapter presents a new scheduling algorithm based on a previous algorithm (GOEL; MARINISSEN, 2003a) used for dedicated TAMs. The proposed algorithm allows co-optimization of wrappers and scheduling, generating test architectures optimized for both SoC test length and silicon area for DfT logic. This algorithm is a first step toward a general NoC-reuse method since it does not require a cycle-accurate NoC model as previous papers (COTA; LIU, 2006). Finally, we compare the proposed algorithm with another algorithm (GOEL; MARINISSEN, 2003a) used for dedicated TAMs.

## 8.1 Problem Statement

The wrapper optimization step generates all information to build a wrapper, however, the test scheduling only needs the test length for each wrapper. Figure 8.1 illustrates the process where a set of wrappers (for instance, this set of wrappers for core $g_1$ is represented by $w_1g_1$ to $w_{c/2}g_1$) is used to build a Pareto graph like the one illustrated in right part of Figure 8.1. This graph represents the wrapper information loaded by the test scheduling tool.

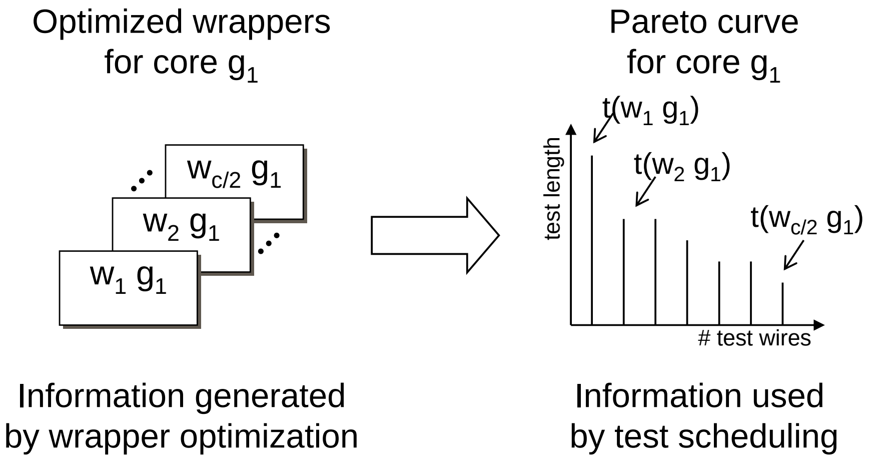


Figure 8.1: Building Pareto curve for the test scheduling. $t()$ represents the test length of a wrapper.

The following problem statement represents the proposed test scheduling algorithm.

**Problem 2** [Test Scheduling for NoC-Based Chips]
Given:

- a graph $G$ which defines the SoC where for each $g \in G$ there is a Pareto curve $P_g$;
- the maximal number of test wires $w_{max}$;
- the physical channel width $c$, in bits;
- the routing algorithm $r()$ used by the NoC.

Determine:

1. the sets of vTAMs;
2. the SoC test length, in clock cycles.

Such that:

*i.* the SoC test length and the silicon area for DfT are minimized while $w_{max}$ is not exceeded and the width of each vTAM is $\leq c/2$.

□

Figure 8.2 graphically presents this problem.

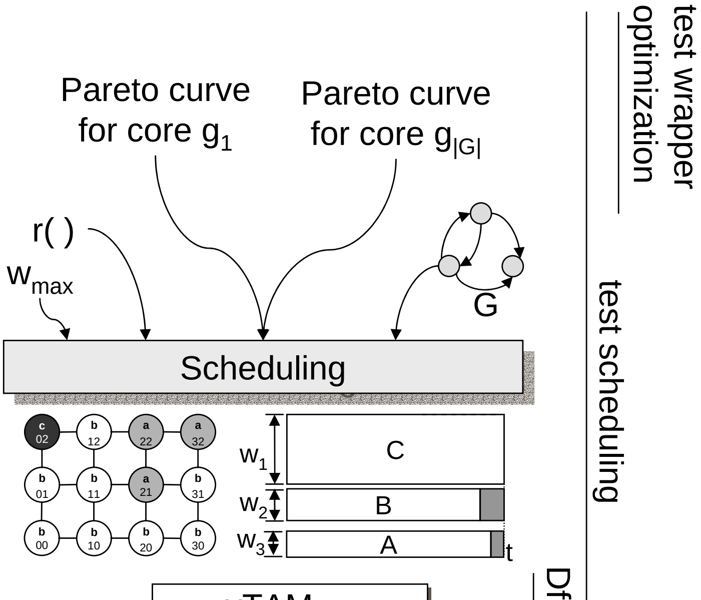


Figure 8.2: Scheduling design flow.

## 8.2 Proposed Test Strategy

### 8.2.1 Previous Test Scheduling Algorithm

Choosing the algorithm for comparison was our first step. The usual SoC test benchmark cannot be used in this comparison due to the lack of some important information like NoC topology, routing algorithm, router test lenght, and standard ports for cores. The comparison can only be accomplished if the original test scheduling algorithm for dedicated TAM is re-implemented and run under an adequate input system, i.e. a system with NoC information. Thus, we have established the follow criteria to choose the test scheduling algorithm:

- *Based on a simple test architecture*:
  The algorithm should be based on fixed-width TAMs and with no preemption support. It is know that these features complicates the hardware design. We want to avoid these complications in first instance;
- *Based on well-defined DfT modules*:
  The algorithm should be compatible with a well defined and established test wrapper design and optimization;
- *Easy to implement algorithm and clear documentation*:
  We need an algorithm simple to be designed since we would need to re-implement it due to the lack of standard benchmarks for NoC-based systems.

The selected algorithm for the comparison and to be adapted to NoC reuse is the one presented in (GOEL; MARINISSEN, 2003a). We initially use only one algorithm due to the need to re-implement the original test scheduling, once there is no subtle benchmark.

We adopted the same terminology used in (GOEL; MARINISSEN, 2003a) to ease the comparison and readability. The main procedure executes the following sub-procedures in sequence: `CreateStartSolution`, `OptimizeBottomUp`, `OptimizeTopDown`, and `Reshuffle`. Figure 8.3 illustrates these optimization algorithms.

The procedure `CreateStartSolution` creates the initial test architecture such that it sorts the modules of the SoC in decreasing order of test data volume, then it assigns the ordered modules to one-bit TAMs. If there are more modules than TAM wires, then, it assigns the remaining modules to the TAM with shortest test length, otherwise, if there are more TAM wires than modules, then, it assigns test wires to the TAM with longest test length. The procedure `OptimizeBottomUp` tries to merge the TAM with shortest test length to another TAM such that the freed test wires are allocated to the longest TAM to reduce the SoC test length. The procedure `OptimizeTopDown` tries to merge the TAM with longest test length to another TAM such that the resulting TAM receives the test wires of both original TAMs. If the resulting TAM has shorter test length than the current SoC test length, then the merge is approved. The procedure `Reshuffle` tries to reduce the current SoC test by moving one module of the TAM with longest test length to another TAM.

### 8.2.2 Employed Data Structure

This section shows how the input system $G$ is transformed into the tile-based graph, which is the graph actually partitioned by the test scheduling.

Figure 8.4, left part, represents the input system. This example system has a set of nine routers (r00 to r22) and a set of ten cores (c0 to c9). This graph is reduced

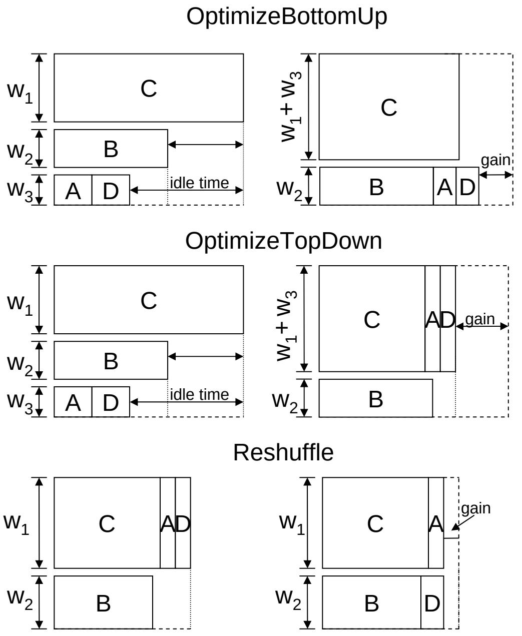


Figure 8.3: Illustrative example of the original optimization algorithms.

to the tile-based graph (depicted in the right part of the figure) by merging the test length of the modules connected to a router. For instance, tile 1 is represented by the test length of the router r02; tile 2 is presented by the sum of test lengths of router r12 and core c0.

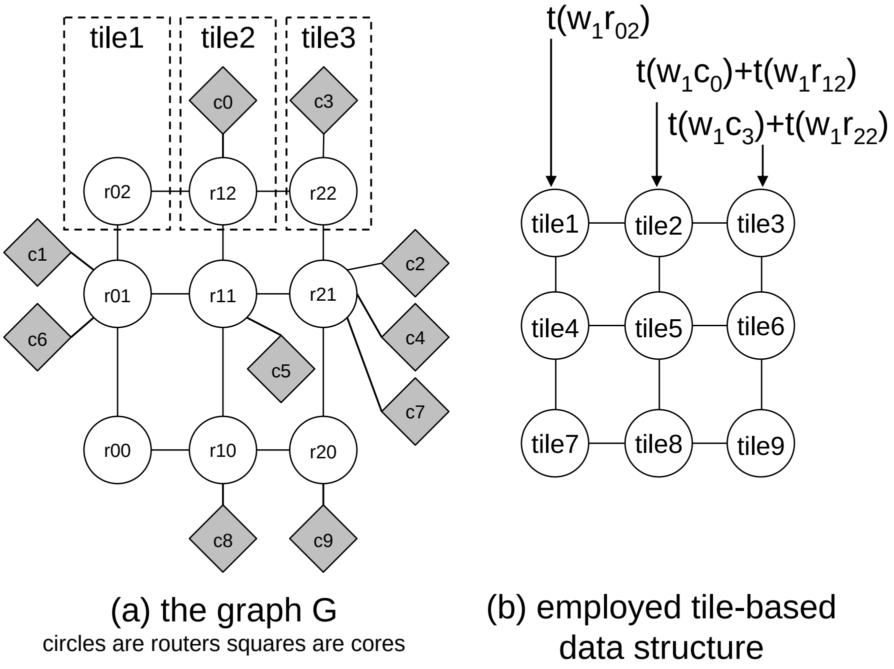
The main property of this approach is that all modules within a tile must be tested by the same TAM. Otherwise, if more than one TAM could test the modules within a tile, there would be packet collision inside the tile once multiple test streams could be activate at the same time. Thus, the tile-based graph is used to avoid packet collision.

### 8.2.3 The Proposed Algorithm

Algorithm 8.1 presents the main procedure for the proposed test scheduling. The procedures in lines 1, 3, 4, 5, and 6 are the ones based on (GOEL; MARINISSEN, 2003a) (we split the original `OptimizeTopDown` algorithm in two parts). There are two constraints that we added to the original algorithms: *the vTAM width constraint and the neighborhood constraint*. Since test data is transported through the NoC channels, *the vTAM width ($w$ in page 77) cannot be higher than the physical channel bandwidth of the NoC*. As explained before, the maximal vTAM width used is half of the physical channel bandwidth ($w \leq c/2$). *The neighborhood constraint is set to avoid packet collision*, i.e. a vTAM consists of neighbor routers to avoid packet collision during test application.

#### *8.2.3.1 Notation*

We use lower case symbols ($x$) to represent a router, upper case symbols ($X$) to represent a set of routers, and slanted upper case symbols ($\mathcal{X}$) to represent a set of sets of routers. $t(R)$ and $w(R)$ are, respectively, the test and width of a set of routers $R$. The term partition is used to refer to the set $\mathtt{R}_{part}$ of a vTAM.



(a) the graph G
circles are routers squares are cores

(b) employed tile-based data structure

Figure 8.4: Transforming the input system in to the employed data structure.

Algorithm 8.1: Main Scheduling Algorithm

```
ModifiedCreateStartSolution;
FixStartSolution;
ModifiedOptimizeBottomUp;
ModifiedOptimizeTopDown1;
ModifiedReshuffle;
ModifiedOptimizeTopDown2;
OptimizeTestWires;
FindATEInterfaces;
```

#### *8.2.3.2 Neighborhood Procedures*

Let us first introduce the sub-procedures `unconnectedRouters`$(R, r)$, `neighborRouters`$(r)$, `neighborTAMs`$(R)$, `bridgeRouters`$(R_1, R_2)$, and `TestTime`$(R, w)$ used in the procedures presented in Section 8.2.3.3.

The procedure `unconnectedRouters`$(R, r)$ has an optional parameter $r$ which is a reference router. When $r$ is not informed, then the first router[1] in the set $R$ is used as reference. The procedure returns the subset of routers in $R$ which the router $r$ cannot have access according to the routing algorithm $r()$. For instance, let us use the system in Figure 8.5(a) and assume XY routing algorithm, then `unconnectedRouters`$(b, 11) = \{20\}$ because the routers 21 and 10 belong to another vTAM, then the router 11 does not have access to the router 20. `unconnectedRouters`$(e) = \emptyset$ because the first router in $e$ has access to the other routers of vTAM $e$.

The procedure `neighborRouters`$(r)$ returns the set of routers directly connected to the router $r$. For example, according to Figure 8.5(a), `neighborRouters`$(12) = \{02, 11, 22\}$.

[1] The routers are ordered in increasing order of test data volume.

The procedure `neighborTAMs`$(R)$ returns the set of partitions directly connected to the partition $R$. For example, according to Figure 8.5(b), `neighborTAMs`$(d) = \{a, b\}$ and `neighborTAMs`$(c) = \{b, a\}$. `neighborTAMs`$(r)$ returns the set of partitions directly connected to the partition $R$ such that $r \in R$. For instance, assuming Figure 8.5(b), `neighborTAMs`$(32) =$ `neighborTAMs`$(d) = \{a, b\}$.

The procedure `bridgeRouters`$(R_1, R_2)$ returns the set of routers in partition $R_1$ in the border with the partition $R_2$. Taking Figure 8.5(b) as an example, `bridgeRouters`$(b, a) = \{11\}$ and `bridgeRouters`$(a, b) = \{10, 21\}$.

The procedure `TestTime`$(R, w)$ calculates the test length according to Equation 5.3, in page 92, of the vTAM $R$ assuming $w$ test wires.

#### *8.2.3.3 Main Procedures*

The procedure `ModifiedCreateStartSolution` has the same logic as the original one, but the vTAM width constraint is checked whenever test wires are added to the longest TAM. If the constraint is met, it means that the maximal number of test wires for this vTAM has been reached and it is not possible to have further optimization. Then the procedure stops and the remaining test wires are not used.

The procedure `FixStartSolution` connects the routers to form vTAMs such that all routers belong to only one vTAM. The Algorithm 8.2 complements the `ModifiedCreateStartSolution` since it implements the neighborhood constraint. Figure 8.5 illustrates the transformation realized by `FixStartSolution`. Note in Figure 8.5(a) that partitions $a$, $b$,$c$, and $d$ have unconnected routers. On the other hand, Figure 8.5(b) has no unconnected router.

Algorithm 8.2: FixStartSolution

```
1 for all $R \in \mathcal{R}${
2     for all $r_{unc} \in$ unconnectedRouters$(R)${
3         find $R_{min}$ for which $t(R_{min}) = \min_{J \in \texttt{neighborTAMs}(r_{unc})} t(J)$;
4         $R_{min} := R_{min} \cup \{r_{unc}\}$ ; $R := R - \{r_{unc}\}$ ;
5         $t(R_{min}) :=$ TestTime$(R_{min}, w(R_{min}))$;
6         $t(R) :=$ TestTime$(R, w(R))$;
7     }
8 }
```

The variable $\mathcal{R}$ in the Algorithm 8.2 represents all sets of vTAMs of the system. For all vTAMs defined in `ModifiedCreateStartSolution`, the procedure `unconnectedRouters` returns the set of routers in $R$ which is not connected. The next step it finds the vTAM $R_{min}$ among the set of neighbor TAMs of the unconnected router $r_{unc}$ with the minimal test length (line 3). Then, the unconnected router $r_{unc}$ is moved from the vTAM $R$ to the $R_{min}$ (line 4) and the test lengths of $R$ and $R_{min}$ are updated (lines 5 and 6). This algorithm ensures that all routers are connected to a vTAM and that there is no unconnected vTAM. Let us assume the system illustrated in Figure 8.5(a). Let us say that $R = b$, then, `unconnectedRouters`$(b) = r_{unc} = \{20\}$. `neighborTAMs`$(r_{unc}) = \{a, c\}$, however, $R_{min} = \{a\}$. Finally, the router 20 is moved from the vTAM $b$ to $a$ since this vTAM is the neighbor vTAM with the shortest test length.

The procedures `ModifiedOptimizeBottomUp`, `ModifiedOptimizeTopDown1`, and `ModifiedOptimizeTopDown2` have similar modifications. They check the vTAM width constraint whenever test wires are added to a vTAM, but they also check

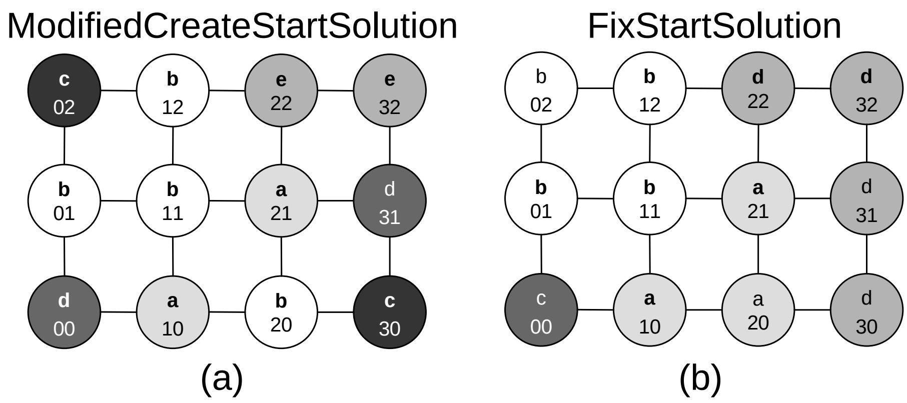


Figure 8.5: Example of the `FixStartSolution`.

the neighborhood constraint by allowing merge candidate vTAMs connected to the focused vTAM.

The procedure `ModifiedReshuffle` tries to move one router in the vTAM with the longest test length to another vTAM to improve the SoC test length. Conversely to the original algorithm, the one presented in Algorithm 8.3 has additional constraints.

Algorithm 8.3: ModifiedReshuffle

```
1  $improve$ := true;
2  while ($improve$){
3      find $R_{max}$ for which $t(R_{max}) = \max_{J \in \mathcal{R}} t(J)$;
4      $TAMFound$ := false;
5      if ($|R_{max}| = 1$) then { $improve$ := false; }
6      else{
7          while ($R_{cand} \in$ neighborTAMs($R_{max}$) $\wedge \neg$ $TAMFound$) {
8              $R_* :=$ bridgeRouters($R_{max}, R_{cand}$);
9              while ($r_* \in R_* \wedge \neg$ $TAMFound$){
10                 $R_{max2} := R_{max} - \{r_*\}$ ; $w(R_{max2}) := w(R_{max})$;
11                 $t(R_{max2}) :=$ TestTime($R_{max2}, w(R_{max2})$);
12                 $R_{cand2} := R_{cand} \cup \{r_*\}$ ; $w(R_{cand2}) := w(R_{cand})$;
13                 $t(R_{cand2}) :=$ TestTime($R_{cand2}, w(R_{cand2})$);
14                 if ($t(R_{cand2}) <$ TestTime($R_{max}$)) then {
15                     if (unconnectedRouters($R_{max2}$) $= \emptyset$) then {
16                         $TAMFound$ := true;
17                         $R_{max} := R_{max2}$; $R_{cand} := R_{cand2}$;
18                         $t(R_{max}) :=$ TestTime($R_{max}, w(R_{max})$);
19                         $t(R_{cand}) :=$ TestTime($R_{cand}, w(R_{cand})$);
20                     }
21                 }
22             }
23         }
24         $improve$ := $TAMFound$;
25     }
26 }
```

The algorithm finds the vTAM $R_{max}$ with the longest test length (line 3). If $R_{max}$ has only one router, then it is impossible to move one item (line 5) and the procedure is finished. Otherwise, for all vTAMs $R_{cand}$, neighbor of $R_{max}$, it finds the set of routers of $R_{max}$ connected to the vTAM $R_{cand}$ (line 8). These routers in the border of $R_{max}$ and $R_{cand}$ are the ones that can be moved, however, some constraints must be checked first. It checks if moving a router $r_* \in R_*$ does not increase the SoC test length (lines 10 to 14). If the move is accepted in terms of test length, then it checks if moving the router $r_*$ will not unconnect the vTAM $R_{max}$ (line 15). If not, the move is accepted and the new SoC test length is updated (lines 16 to 19).

For example, considering the system in Figure 8.5(b) and assuming that partition $d$ correspond to $R_{max}$. Then, $R_{cand}$ might assume $a$ and $b$. When $R_{cand} = a$, $\texttt{bridgeRouters}(R_{max}, a) = \{22, 30, 31\}$ while when $R_{cand} = b$, $\texttt{bridgeRouters}(R_{max}, b) = \{22\}$. Let us assume that when it moves the router 31 and 22 the test length reduces. However, when the router 31 is moved the router 30 will be unconnected, which is not allowed. Finally, the router 22 is moved to the partition $b$ because it passes in all constraints and it reduces the SoC test length.

The procedure `OptimizeTestWires` described in Algorithm 8.4 tries to optimize the number of test wires. Once there is no additional improvement in test length to be done, it iteratively tries to reduce the number of test wires of all vTAMs (line 6) under the condition that the current SoC test length is not exceeded (lines 8 and 9). By reducing the number of test wires, it also reduces the number of test pins, the memory requirement in the tester, and the silicon area to implement the DfT presented in Section 4.8, page 80. Figure 8.6 illustrates the optimization. One can realize that even the vTAM with the longest test length can be optimized because increasing the number of test wires of a core not necessarily decrease its test length due to the Pareto behavior (GOEL; MARINISSEN, 2003a).

Algorithm 8.4: OptimizeTestWires

```
1  $T := \max_{J \in \mathcal{R}} t(J)$;
2  for all $R \in \mathcal{R}\{$
3      improve := true;
4      while (improve){
5          if $(w(R) > 1)$ then {
6              $R_* := R; w(R_*) := w(R) - 1$;
7              $t(R_*) := \texttt{TestTime}(R_*, w(R_*))$;
8              if $(t(R_*) > T)$ then {improve := false;}
9              else{ $w(R) := w(R_*)$; $t(R) := t(R_*)$; }
10         }else{ improve := false; }
11     }
12 }
```

The procedure `FindATEInterfaces` finds the minimal number of ATE interfaces ($\mathtt{R}_{atei}$ in page 77) such that all modules in a vTAM can be reached from an ATE interface. For example, ATE interfaces of the partition $b$ in Figure 8.7 are $\{11, 30\}$, while for partition $a$ is only $\{22\}$. Algorithm 8.5 presents the pseudocode of `FindATEInterfaces`.

Let us initially define the variable $v$ as a set of the tuple $\{r, U\}$, where $r$ is a router and $U$ is a set of routers that cannot be accessed by $r$. For instance, the value of $v$ to the vTAM $b$ in Figure 8.7 (assuming $r() = \mathrm{XY}$) is

$$v = \{\{12, \{01, 20, 30, 31\}\}, \{11, \{20, 30, 31\}\}, \{01, \{12, 10, 20, 30, 31\}\}, \{10, \{01, 31\}\}, \ldots\}$$

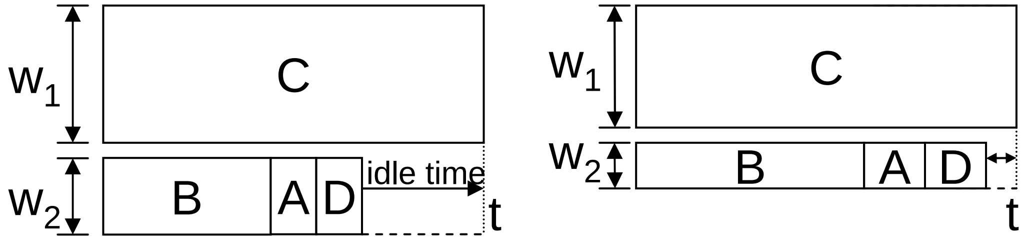


Figure 8.6: Example of optimization caused by `OptimizeTestWires`. Note that the test length is the same but the number of test wires has been reduced in $w_1$ and $w_2$.

Algorithm 8.5: FindATEInterfaces

```
1  for all $R \in \mathcal{R}\{$
2      $minSolution := \infty;$
3      for all $r_* \in R\{$
4          $v_*.r := r_*;$
5          $v_*.U :=$ unconnectedRouters$(R, r_*);$
6          $v := v \cup \{v_*\};$
7      }
8      for all $v_1 \in v\{$
9          $R_{unc} := v_1.U; R_{ate} := \{v_1.r\};$
10         do {
11             $minSize := \infty;$ $minRouter := -1;$
12             for all $v_{cand} \in v\{$
13                 if $(v_{cand}.r \ni R_{ate})$ then {
14                     $U_* := R_{unc} - v_{cand}.U;$
15                     if $(|U_*| < minSize)$ then {
16                         $minSize := |U_*|;$ $minRouter := v_{cand}.r;$
17                     }
18                 }
19             }
20             if $(minRouter \neq -1)$ then {
21                 find $v_{min}$ for which $\forall j \in v, j.r = minRouter;$
22                 $R_{unc} := R_{unc} - v_{min}.U; R_{ate} := R_{ate} \cup \{v_{min}.r\};$
23             }
24         }while $(R_{unc} \neq \emptyset);$
25         if $(|R_{ate}| < minSolution)$ then {
26             $R_{min} := R_{ate};$ $minSolution := |R_{ate}|;$
27         }
28     }
29     $\mathcal{R}_{sol} := \mathcal{R}_{sol} \cup \{R_{min}\};$
30 }
```

For all vTAMs $R$ of the system it defines $v$ (lines 3 to 6) and finds the minimal number of ATE interfaces for the vTAM $R$ (loop begins at line 8). $R_{ate}$ is the current set of ATE interfaces of the vTAM $R$, while $R_{unc}$ is the set of unconnected routers (line 9), i.e. the routers in $R$ that are not accessible by the ATE interfaces in $R_{ate}$. The loop beginning at line 12 iterates over $v$ again to find the other ATE interfaces of $R$. If the element $v_{cand}$ is not yet included in the set of ATE interfaces (line 13), it evaluates if the number of not reachable routers is the minimal found so far (*minSize* in line 15). If the current set of unconnected routers is the minimal,

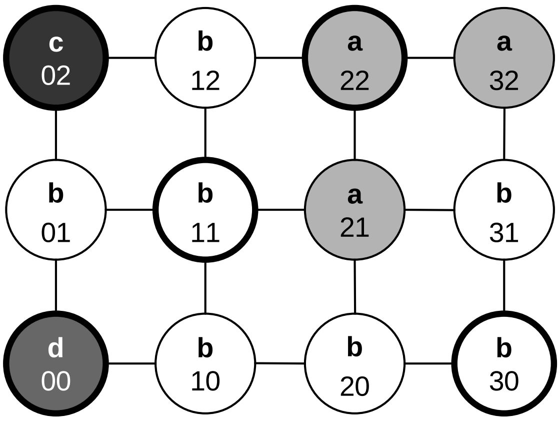


Figure 8.7: Example of a partitioned NoC. The letters identify the partitions, the numbers refer to the routers, and the circles with fat lines indicated where the ATE interfaces are located.

then the *minSize* is updated and the current best router candidate to ATE interfaces is also update (*minRouter* in line 16). In line 21 it finds the $v$ element corresponding to the *minRouter*, it includes its router to the set of ATE interfaces $R_{ate}$ and excludes the routers that can be reached from $v_{min}.r$ from the set of unreachable routers $R_{unc}$ (line 22). This procedures repeats until all routers are reachable, i.e. $R_{unc} = \emptyset$ (line 24). However, the solution found in the *do* loop might be different depending on the order of the elements in $v$. For this reason, the outer loop beginning at line 6 executes the whole *do* loop again, but having a new $v$ element as a starting element (line 9). For the same reason, it tests in line 25 if the solution just found by the *do* loop is the absolute minimal found so far for the vTAM $R$ (*minSolution*). At line 26, $R_{min}$ has the absolute minimal number of ATE interfaces for the vTAM $R$. $\mathcal{R}_{sol}$ stores the sets of ATE interfaces for all $R \in \mathcal{R}$ (line 29).

### 8.2.4 Minimizing Test Length and Silicon Area

The main optimization criterion of the proposed algorithm is test length. Once test length cannot be reduced, silicon area for DfT is optimized. The algorithm *OptimizeTestWires* reduces the number of test wires in each partition because the number of test wires has influence on the DfT silicon area (AMORY et al., 2007). Moreover, the algorithm *FindATEInterfaces* finds the minimal number of ATE interfaces, which also cost silicon area.

## 8.3 Results

### 8.3.1 Experimental Setup

We need systems with NoC information to proceed with the comparison between conventional scheduling and the proposed method, which are not available in the literature. For instance, the cores in the ITC'02 benchmarks do not have functional protocol ports required by our method. The ITC'02 benchmarks have neither NoC topology nor NoC placement. The ITC'02 benchmarks also do not have test information for routers.

We could not find in the literature the test length results based on conventional scheduling for these kind of systems (or any similar NoC-based system). Thus, we re-implemented the scheduling algorithm presented in (GOEL; MARINISSEN, 2003a) and its wrapper optimization algorithm described in (MARINISSEN; GOEL; LOUSBERG, 2000) to do the comparison (this implementation was validated with the results presented in (GOEL; MARINISSEN, 2003a)).

### 8.3.2 Introduction to Relevant Variables

The size of the *channel width* determines the maximal width of the 'virtual TAM'. The channel width also determines the kind of port used by the modules (cores and routers). The data width of the port must match the channel width, i.e. channel width of 32 bits requires ports with 32-bit data terminals. The same also applies for routers.

The *router weight* (number of test patterns of the routers) determines the amount of the SoC test length spent to test routers. For instance, lower router weight means that the routers are tested faster and a small portion of the SoC test time is spent on testing the NoC.

Recall that these variables are given in actual designs, but we investigate if these two variables have any influence on the comparison between the conventional and the proposed approach.

### 8.3.3 Experiment Model

Let $S$, $C$, $R$, $W$, $P$, and $D$ be, respectively, the set of SoCs, channel widths, router weights, TAM widths, NoC placements, and NoC dimensions (a pair XY) used in this paper. All experiments assume the XY routing algorithm ($r()$). Except by $P$, the other sets are defined below.

$$\begin{aligned}
S &= \{d695, f2126, g1023, h953, p22810, p34392, p93791, q12710, t512505, u226\} \\
C &= \{32, 64, 128\} \\
R &= \{low, medium, high\} \\
W &= \{16, 24, 32, 40, 48, 56, 64\} \\
D &= \{(3,3), (2,2), (4,4), (3,3), (5,5), (4,4), (6,6), (2,2), (6,6), (3,3)\}
\end{aligned}$$

An experiment is defined as a tuple $\{s, c, r, w, p_s, d_s\}$ such that $s \in S$, $c \in C$, $r \in R$, $w \in W$, $p_s \in P$, and $d_s \in D$. The sets $P$ and $D$ are associated to the set $S$. For instance, d695 uses the first placement $(3,3)$, f2126 uses the second one $(2,2)$, and so on.

The set $P$ has a set of ten placements randomly generated for each system such that a placement can have more than one core per router and it is also possible to have a router without core. This set of placements is used in all experiments.

Figure 8.8 shows how an experiment is set up for each system. This flow ensures that both conventional and proposed scheduling tools use the same core and router configuration. Once the modules were generated, they are optimized by their respective wrapper optimization algorithm for finally execute the test scheduling. The NoC reuse problem also needs NoC information like placement and dimension.

The procedure *CoreGen*$(c)$ sets the correct terminal count for the set of cores of a SoC. The number of terminals of a core is determined by *itc02* $+2 \times (39 + c)$, where *itc02* is the original terminal count for the core as defined in the ITC02 SoC benchmark. The rest of the expression $(2 \times (39 + c))$ represents the number of terminals of the core related to the input and output ports illustrated in Figure 8.9.

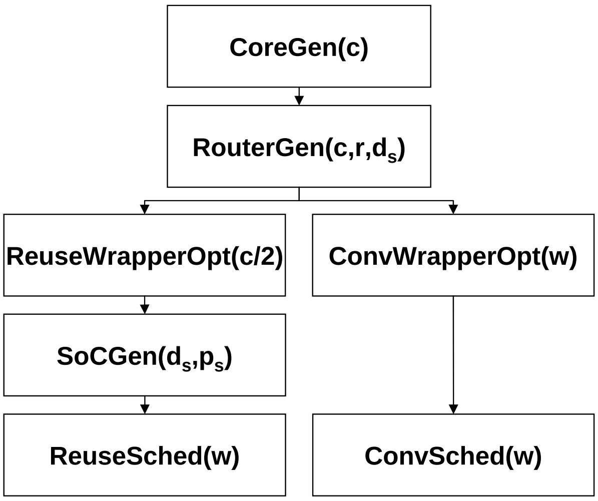


Figure 8.8: Flow of an experiment for each SoC.

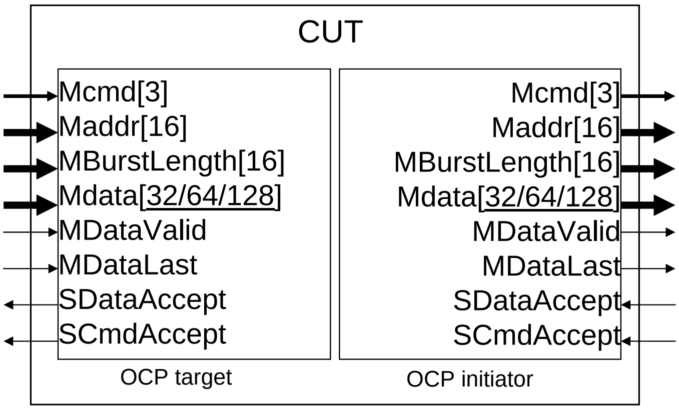


Figure 8.9: OCP port used in the experiments.

Each port has 39 control terminals and $c$ data terminals. For instance, assuming that the original terminal count of a core is 100, $c = 128$, then the total number of terminals of the core is $100 + 2 \times (128 + 39) = 434$ terminals.

The procedure $RouterGen(c, r, d_s)$ sets the number of routers, the number of test patterns, and the terminal count of a router. For sake of simplicity we assumed that all routers in a system are the same, i.e. they have five bi-directional ports and the same number of test patterns. We assume a router port can have 32, 64, or 128 data terminal, and two control terminals (e.g. request and acknowledge terminals). Thus, the number of terminals of a router is $5 \times (2 \times (2 + c))$. For instance, if $c = 32$, then the router has $5 \times 2 \times (2 + 32) = 340$ terminals. Moreover, the router has 50 scannable flip-flops related to internal control logic.

The procedure $ReuseWrapperGen(c/2)$ optimizes the wrapper for both cores and routers based on the algorithm presented in (AMORY et al., 2007) for NoC reuse wrappers, while the procedure $ConvWrapperGen(w)$ also optimizes wrappers for both cores and routers, but using the algorithm (MARINISSEN; GOEL; LOUSBERG, 2000) used for conventional wrappers. Both the conventional and the proposed wrapper optimization tools were implemented to support both soft and hard cores because we assume that all cores are hard-cores, while the routers are soft-cores. We believe that NoCs are implemented as soft-core because, on the other hand, if the NoC is implemented as a hard-core, it would loose configurability, which are one of the main benefits of NoCs.

The procedure $SoCGen(d_s, p_s)$ generates ten NoC-based SoCs for the system $s$, where each SoC has dimension $d_s$ and placement $p_{s_i}$, where $i = \{1, \ldots, 10\}$.

The procedure $ReuseSched(w)$ executes the proposed and the conventional test scheduling algorithm, respectively.

### 8.3.4 Definitions

Let $t_c(s, w)$ be defined as the conventional test length of the SoC $s$ with $w$ test wires, where $s \in S$ and $w \in W$. Likewise, let us define $t_{r_{mean}}(s, w)$ as the mean reuse test length for ten different placements of the SoC $s$ with $w$ test wires.

Let us define $g_{s,w} = \frac{t_{r_{mean}}(s,w)*100}{t_c(s,w)}$ as the gain of the mean reuse test length $t_{r_{mean}}()$ compared to the conventional test length $t_c()$ of the system $s$ with $w$ test wires . When $g_{s,w} < 100$ it means that the mean reuse test length is lower than the conventional test length.

### 8.3.5 Pruning Search Space

The test scheduling problem based on NoC reuse involves more variables than conventional TAMs like channel width, router weight, and placement. *Combining these additional variables with the common ones used in conventional scheduling ($w_{max}$ and the SoC) generates a huge search space.* For instance, all combinations of the tuple $\{s, c, r, w, p_s, d_s\}$ would require $|S| \times |C| \times |R| \times |W| = 10 \times 3 \times 3 \times 7 = 630$ executions of the conventional and $630 \times 10 = 6300$ executions of the proposed scheduling (10 different placements).

The goal of this section is to study the impact of channel width, router weight, and placement on the proposed test scheduling. Based on this study we can prune the search space by concluding which are the most significant variables.

#### *8.3.5.1 Evaluating the Impact of the Channel Width*

Figure 8.10(a) illustrates the result of the channel width ($C$) evaluation. This experiment is set up with $r =$*medium* and $w = 16$. The free variables are $S$ and $C$ such that the x axis in Figure 8.10(a) represents the pairs of $S \times C$.

The problem of this experimental setup is that, for example, the first core of h953-32 has 406 terminals while the same core of h953-128 has 598 terminals since $c$ is different for each case. The last set up would obviously require more test length than the first one. To overcome this problem we have set the experiment such that, for example, the first core of the SoCs h953-32, h953-64, and h953-128 have 598 terminals.

The data in Figure 8.10(a) is calculated such that each point in the graph is $g_{s,16}$ where $s \in S \times C$, i.e. the gain of the reuse over the conventional considering 16 test wires.

Two patterns can be identified in this figure. First, for some systems, identified by the dashed circles, there are an upward pattern, which means that the reuse approach gets worse as the channel width is reduced. The second case, identified by the circle with continuous line, has a constant pattern, which means that the channel width has no influence on the relative test length.

This evaluation says that channel width has no influence on test length when the

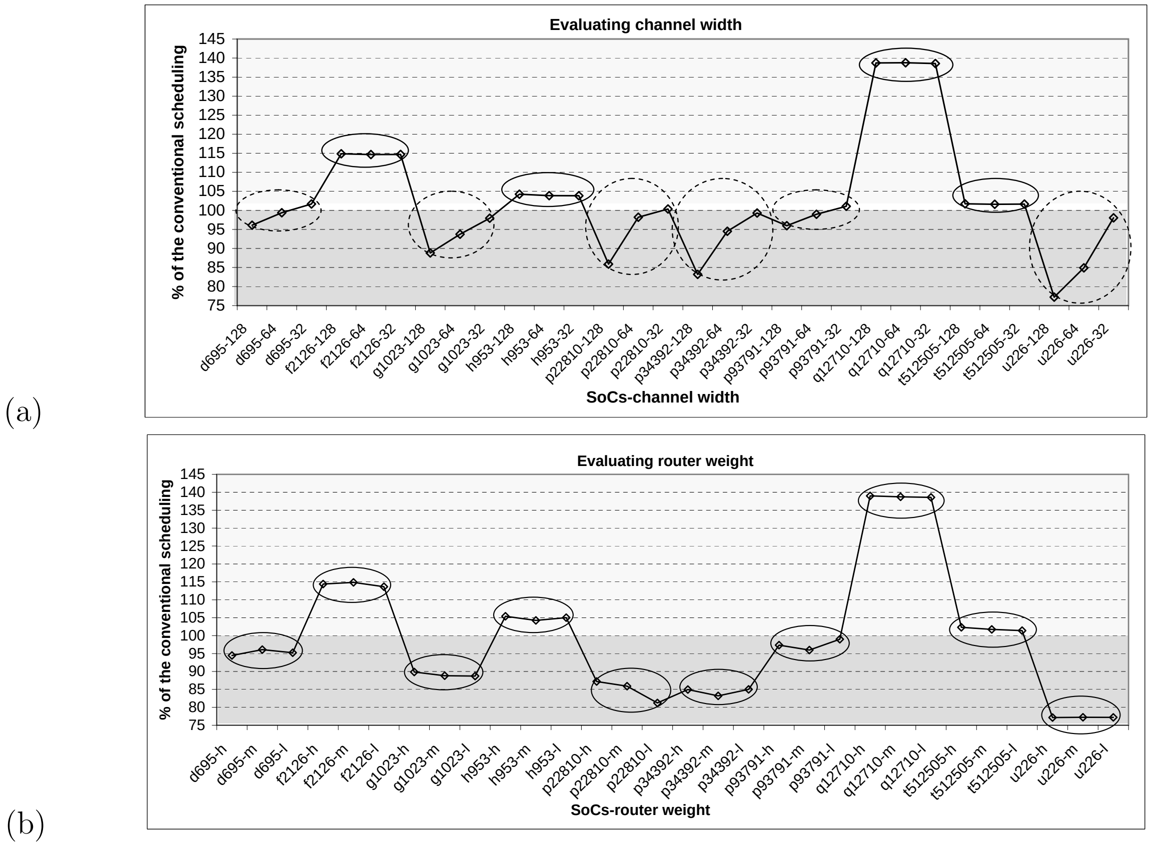


Figure 8.10: Evaluating the impact of the channel width (a) and the router weight (b) on the SoC test length. The circles help to identify the values for the same SoC. The dashed circle identifies the systems with upward pattern while the other circles identify the constant pattern. The values in the shadow area mean that the proposed approach is faster on average. SoC set up for (a) is $r = medium$ and $w = 16$ and for (b) is $c = 128$ and $w = 16$.

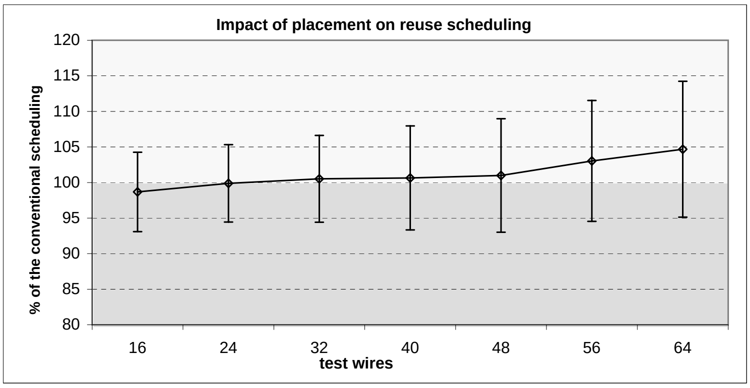


Figure 8.11: Impact of the placement on SoC test length. The std. dev. bar over the mean gains shows the possible test length variation caused by ten different placements. SoC set up: $c = 128$ and $r =$*medium*.

SoC have bottleneck cores, otherwise, the reuse approach performs better with wider channels. It can be explained by the Equation 5.5, page 92, for core test length. For instance, let us assume that the scan-in length $s_i = 100$, channel width $c = 32$, $w = 3$ test wires, which resulted in $t_i = 91$ clock cycles. However, if we change the channel width to $c = 64$, then $p_i = \left\lfloor \frac{64}{3} \right\rfloor = 21$ and $t_i = \left(\left\lceil \frac{100}{21} \right\rceil - 1\right) \times 21 + 1 = 85$ clock cycles. This result indicates that the wider is the channel, the shorter will be the core test length. The combination of shorter core test lengths result in a shorter SoC test length.

*8.3.5.2 Evaluating the impact of the Router Weight*

We set up the router weight experiment, shown in Figure 8.10(b), to $c = 128$, and $w = 16$. The free variables are $S$ and $R$ such that the x axis in Figure 8.10(b) represents the pairs of $S \times R$. Similar results were also observed for other SoC set ups. The data in Figure 8.10(b) was calculated likewise Figure 8.10(a).

*8.3.5.3 Evaluating the Impact of the Placement*

This experiment is set up as $c = 128$ and $r =$*medium*. $c = 128$ has been chosen because it allows more test wires (up to 64) for the scheduling tool. Since router weight has no relevant impact on test length, $r =$*medium* has been used. $S$ and $W$ are the free variables.

The curve in Figure 8.11 is calculated according the mean gain for all SoCs. For instance, point related to 16 test wires is calculated as $\frac{g_{d695,16} + \cdots + g_{u226,16}}{10}$.

Figure 8.11 presents the impact of placement on the test length. It includes standard deviation bars to show the possible variation of the overall mean gain with the placement. This figure also tells that the standard deviation is bigger when more test wires are used (compare the size of the error bar for 64 and for 16 test wires).

### 8.3.6 Main Results

The main experiment is set up as in Section 8.3.5.3. Figure 8.12 illustrates the same mean gain presented in Figure 8.11, but we removed the error bars and included the median.

We observed that the overall mean gain shown in Figure 8.12 and 8.11 is negatively affected by some systems like q1271 with high gain deviation, increasing the

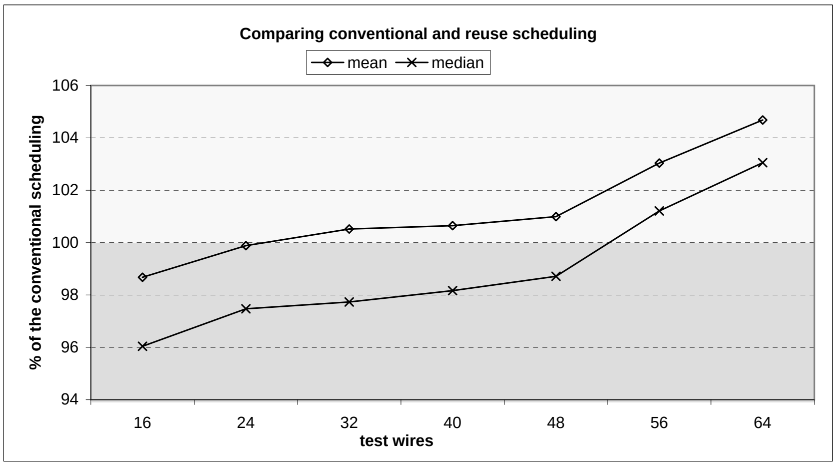


Figure 8.12: Comparison of test length between conventional and NoC reuse scheduling (expressed both in mean and median gains). The values in the shadow area means that the proposed approach is faster on average. SoC set up: $c = 128$ and $r = medium$.

overall mean gain. For this reason, the median of gains is also plotted in the graph, which is a better representative number.

Observing the median gain in Figure 8.12 it can be observed that the reuse test scheduling can be faster than the conventional approach. This graph also tells that the reuse approach usually performs better with less number of test wires.

On the other hand, the values presented in Figure 8.12 are just mean and median values over different placements. In an actual SoC, the placement is given. Thus, this result tells at least that the reuse approach has competitive test length compared to the conventional approach.

Table 8.1 is an extended version of data presented in Figure 8.12. The table presents the SoC evaluated, the number of test wires ($w$), the test length of the conventional scheduling tool ($t_c$), the mean test length of the proposed scheduling for NoC reuse ($t_{r,mean}$), and the reuse/conv represents the mean gain ($g_{s,w}$) which compares both scheduling approaches.

### 8.3.7 Illustrative Example

Figure 8.13 illustrates the scheduling for both approaches considering the SoC d695 set up to $c = 128$, $r = medium$, and $w = 32$. In the conventional scheduling, the total number of TAMs is 13. Nine of them are used to the routers, while the rest is used for the cores. The resulting SoC test length for the conventional approach is 36859 clock cycles. Figure 8.13 also shows the placement (small numbers refer to routers and big numbers refer to the cores attached to each router), the partitions, and the scheduling used for the proposed approach. The final SoC test length for the proposed approach is 34770 clock cycles.

### 8.3.8 Estimating Wire Length Savings

The main benefit of NoC reuse compared to dedicated TAMs is the reduction of long global wires to implement the dedicated TAMs. An exact measure of the wire length savings is complicated because it would required SoC area and floorplaning information not available in the ITC02 benchmarks. For this reason we propose an approximate method to estimate the wire length saving.

| SOC | w | conv | reuse-mean | reuse/conv | SOC | w | conv | reuse-mean | reuse/conv |
|---|---|---|---|---|---|---|---|---|---|
| d695 | 16 | 71254 | 68475 | 96.10 | p34392 | 16 | 1773610 | 1475641 | 83.20 |
| | 24 | 47872 | 46191 | 96.49 | | 24 | 1155070 | 1004815 | 86.99 |
| | 32 | 36859 | 35598 | 96.58 | | 32 | 883601 | 782077 | 88.51 |
| | 40 | 29974 | 28943 | 96.56 | | 40 | 696556 | 652163 | 93.63 |
| | 48 | 25131 | 24000 | 95.50 | | 48 | 598417 | 609949 | 101.93 |
| | 56 | 21135 | 21125 | 99.95 | | 56 | 551119 | 606232 | 110.00 |
| | 64 | 18173 | 18778 | 103.33 | | 64 | 544579 | 601359 | 110.43 |
| f2126 | 16 | 382275 | 439093 | 114.86 | p93791 | 16 | 2130063 | 2044513 | 95.98 |
| | 24 | 335334 | 384416 | 114.64 | | 24 | 1401929 | 1380268 | 98.45 |
| | 32 | 335334 | 383305 | 114.31 | | 32 | 1056006 | 1044235 | 98.89 |
| | 40 | 335334 | 383162 | 114.26 | | 40 | 851859 | 849902 | 99.77 |
| | 48 | 335334 | 383275 | 114.30 | | 48 | 735157 | 690205 | 93.89 |
| | 56 | 335334 | 382580 | 114.09 | | 56 | 605215 | 601469 | 99.38 |
| | 64 | 335334 | 382488 | 114.06 | | 64 | 529719 | 525372 | 99.18 |
| g1023 | 16 | 94942 | 84317 | 88.81 | q12710 | 16 | 2222349 | 3082996 | 138.73 |
| | 24 | 62913 | 58655 | 93.23 | | 24 | 2222349 | 3084733 | 138.81 |
| | 32 | 47277 | 43228 | 91.44 | | 32 | 2222349 | 3080736 | 138.63 |
| | 40 | 38318 | 35611 | 92.94 | | 40 | 2222349 | 3078188 | 138.51 |
| | 48 | 32543 | 30453 | 93.58 | | 48 | 2222349 | 3076399 | 138.43 |
| | 56 | 28576 | 27635 | 96.71 | | 56 | 2222349 | 3076168 | 138.42 |
| | 64 | 23998 | 25049 | 104.38 | | 64 | 2222349 | 3076066 | 138.42 |
| h953 | 16 | 124135 | 129417 | 104.26 | t512505 | 16 | 10604336 | 10788891 | 101.74 |
| | 24 | 119357 | 125079 | 104.79 | | 24 | 10453470 | 10604673 | 101.45 |
| | 32 | 119357 | 123349 | 103.34 | | 32 | 5289094 | 5641771 | 106.67 |
| | 40 | 119357 | 122781 | 102.87 | | 40 | 5228420 | 5377666 | 102.85 |
| | 48 | 119357 | 122523 | 102.65 | | 48 | 5228420 | 5373960 | 102.78 |
| | 56 | 119357 | 122299 | 102.46 | | 56 | 5228420 | 5373817 | 102.78 |
| | 64 | 119357 | 122196 | 102.38 | | 64 | 5228420 | 5373715 | 102.78 |
| p22810 | 16 | 759188 | 652088 | 85.89 | u226 | 16 | 60014634 | 46338535 | 77.21 |
| | 24 | 508022 | 441846 | 86.97 | | 24 | 40919068 | 31522271 | 77.04 |
| | 32 | 382783 | 339004 | 88.56 | | 32 | 31371285 | 24554574 | 78.27 |
| | 40 | 314216 | 271910 | 86.54 | | 40 | 24608398 | 19328372 | 78.54 |
| | 48 | 257179 | 230688 | 89.70 | | 48 | 21823502 | 16843936 | 77.18 |
| | 56 | 225100 | 204796 | 90.98 | | 56 | 19095564 | 14433285 | 75.58 |
| | 64 | 197198 | 179308 | 90.93 | | 64 | 16367627 | 13251262 | 80.96 |

Table 8.1: Extended results with the SoCs set up to $c = 128$ and $r =$*medium*. reuse/conv < 100 means that the proposed approach is faster.

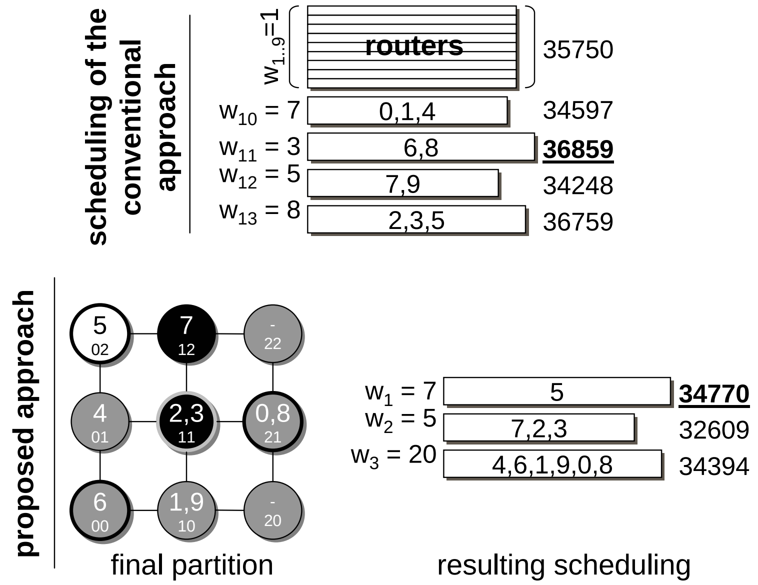


Figure 8.13: Example of both resulting schedulings for the d695 SoC. SoC set up to $c = 128$, $r =$*medium*, and $w = 32$.

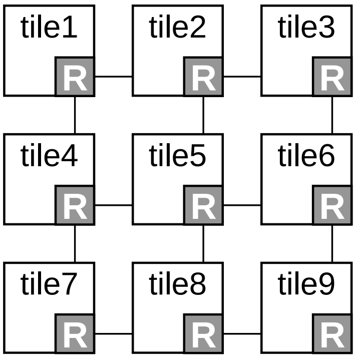


Figure 8.14: NoC-based system modeled as regular tiles are assumed for wire saving estimation.

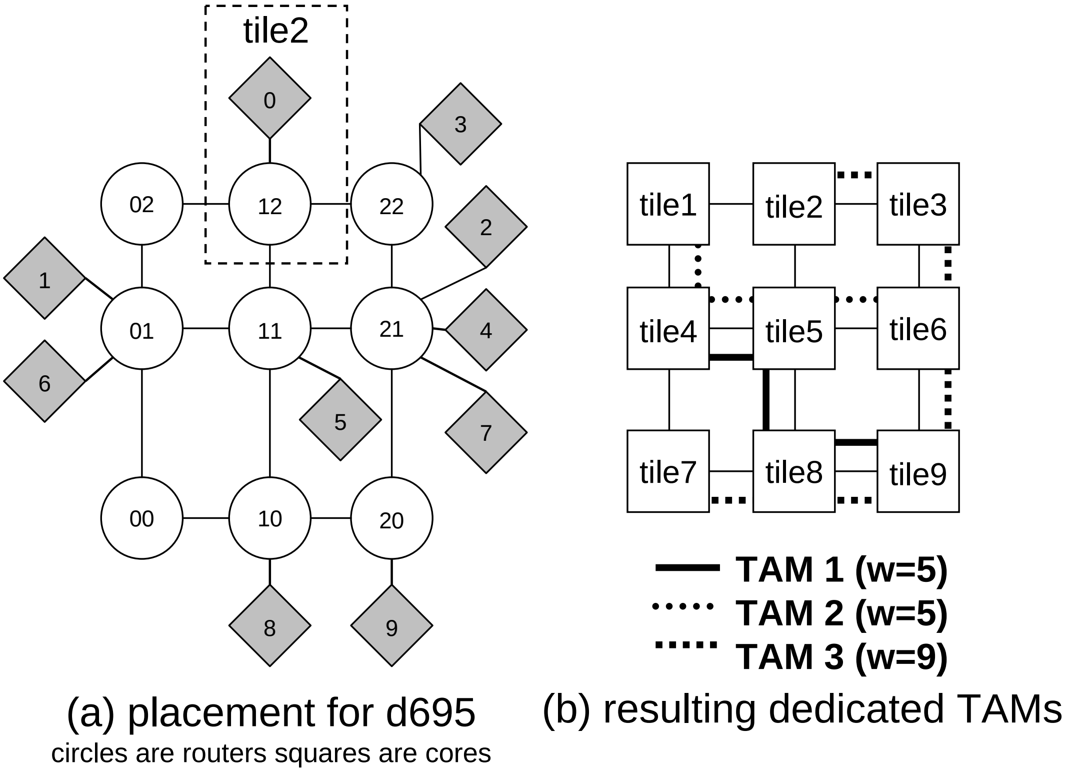
#### *8.3.8.1 Proposed Wire Length Estimation Method*

Let us assume that the SoC is organized in tiles such that there is one tile for each router of the NoC. Figure 8.15 illustrates the proposed tile-based NoC model. The tiles are evenly distributed in the entire SoC area such that the distance between any two neighbor tiles are the same. Each tile can have zero or more cores. The wire length between cores within the same tile is supposed to be zero, while the wire length between cores in different tiles is equivalent to the number of hops between these two tiles.

This simplified NoC model can be used to estimate the amount of wires required to implement a dedicated TAM. The method counts the minimal number of hops required to reach all modules within a dedicated TAM. The number of hops is multiplied by the TAM width and by two to represent the wires for test stimuli and responses. The result represents the number wires, where each wire has length of a hop, required to create a dedicated TAM. Summing up the number of wires for all TAMs of a system gives the total number of wires for the system. Equation 8.1 represents the proposed wiring estimation method, where $n$ is the number of TAMs, $h_i$ is the number of hops to implement TAM$_i$, and $w_i$ is the width of the TAM$_i$.

$$w_{sys} = \sum_{i=1}^{n} h_i \times w_i \times 2 \tag{8.1}$$

Figure 8.15 shows an example of the proposed wire length estimation method. Let us assume the system presented in Figure 8.15(a) and the following resulting core assignment $TAM1 = \{1, 5, 6, 8, 9\}$, $TAM2 = \{4, 01, 11, 12, 02\}$, $TAM3 = \{0, 2, 3, 7, 00, 01, 02, 12, 22\}$ calculated by the conventional scheduling tool assuming 16 test pins. The minimal wire length for this test architecture is presented in Figure 8.15(b). For instance, TAM1 has five cores where two of them are located in the tile 4 and the remaining cores are located in tiles 5, 8, and 9. The minimal distance between these four tiles is three hops (see TAM1 in Figure 8.15(b)). Since the width of TAM1 is five, then it results in 2 × 3 × 5 = 30 wires required to implement the dedicated TAM1. The total wire count for the entire test architecture is 150 wires (30 for TAM1, 30 for TAM2, and 90 for TAM3) where each wire has length of one hop.



(a) placement for d695
circles are routers squares are cores

(b) resulting dedicated TAMs

Figure 8.15: Example of estimated wire length required to create the dedicated TAMs.

#### *8.3.8.2 Wire Length Estimation Model*

The proposed wire length estimation method is equivalent to the minimum rectilinear Steiner tree problem which is $\mathcal{NP}$-complete (PREAS; LORENZETTI, 1988). The problem can be modeled as follows: given $N$ points in the plane, find a minimum length tree of rectilinear edges which connects the points (KAHNG; ROBINS, 1992).

#### *8.3.8.3 Wire Length Estimation Algorithm*

The Algorithm 8.7 estimates the wire length required to implement dedicated TAMs.

The Algorithm 8.6 calculates the half perimeter among a set of points $V$, i.e. the minimal distance between the set of points. The problem is that the distance depends on the path used between two points. For this reason, the Algorithm 8.7 inserts Steiner points, i.e. points that are included in the set $V$ such that it forces the Algorithm 8.6 to follow the path with the actual minimal distance between these points.

A point is a pair $\{x, y\}$. In Algorithm 8.6, $t_{min}, s$, and $t$ represent points; $V, S$, and $T$ are sets of points; $hp, d_{min}$, and $d$ are integer variables.

In Algorithm 8.7, $V, V_{min}, S, V_2$, and $V_{cur}$ are sets of points. $bl$ and $tr$ are points. $cost_{min}$, $cost$, $cost_{cur}$, and $n_{pts}$ are integer variables. The functions `bottomLeft`$(V)$ and `topRight`$(V)$ return, respectively, the bottom-left and the top-right coordinates of the set $V$. These values are used to set the boundary of the search space for Steiner points.

Algorithm 8.6: HalfPerimeter($V$)

```
$S := \{V[1]\}; T := V - \{V[1]\}; hp := 0;$
while $(|T| > 0)\{$
    $d_{min} := \infty; t_{min} := \emptyset;$
    for all $s \in S\{$
        for all $t \in T\{$
            $d := abs(t.x - s.x) + abs(t.y - s.y);$
            if $(d < d_{min})$ then$\{$
                $d_{min} := d; t_{min} := t;$
            $\}$
        $\}$
    $\}$
    $hp := hp + d_{min};$
    $S := S \cup \{t_{min}\};$
    $T := T - \{t_{min}\};$
$\}$
return $hp;$
```

Algorithm 8.7: Minimum Rectilinear Steiner Tree($V$)

```
$V_{min} := V; cost_{min} :=$ HalfPerimeter$(V);$
$bl :=$ bottomLeft$(V); tr :=$ topRight$(V); n_{pts} := 1;$
while (true)$\{$
    $S := \emptyset; V_2 := \emptyset;$
    for 1 to $n_{pts}\{$
        $S := S \cup \{bl\};$
    $cost_{cur} := cost_{min}; V_{cur} := V_{min};$
    while (true)$\{$
        $V_2 := V_2 \cup S;$
        $cost :=$ HalfPerimeter$(V_2);$
        if $(cost < cost_{cur})$ then$\{$
            $cost_{cur} := cost;$
            $V_{cur} := V_2;$
        $\}$
        $finish := 0;$
        //increment the points in S
        if $(finish)$ then break;
    $\}$
    if $(cost_{cur} \geq cost_{min})$ then break;
    $cost_{min} := cost_{cur};$
    $V_{min} := V_{cur};$
    $n_{pts} := n_{pts} + 1;$
$\}$
return $cost_{min};$
```

For example, let us say that we want to determine the minimal distance of the TAM $a$ illustrated in Figure 8.16. First, it sets the search space boundary by setting $bl = 10$ and $tr = 32$. The elements of TAM $a$ are represented by $V = \{10, 22, 31\}$. The Algorithm 8.6 finds the minimal distance between the pair of points in $V$. Thus, in this example the half perimeter between 10 and 31 is 3 either via point 20 or via point 11. Let us assume that the path 10, 20, 30, 31 has been selected. The distance between 31 and 22 is 2 either via point 21 or via point 32. Let us assume that the

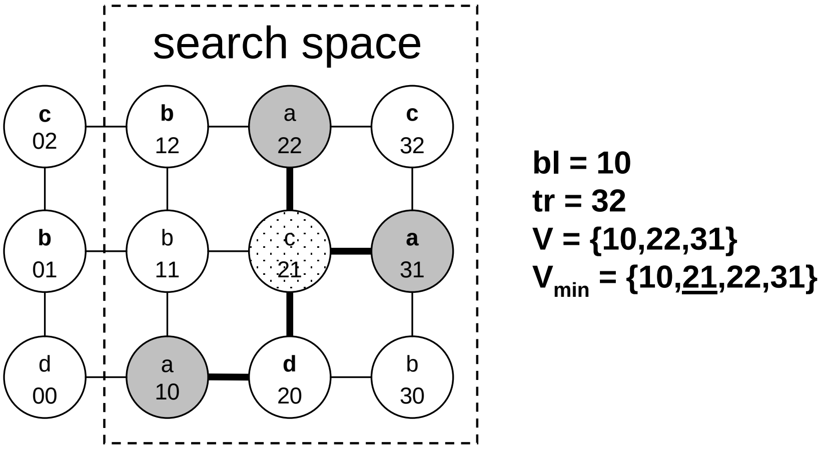


Figure 8.16: Minimum rectilinear steiner tree algorithm. The Steiner point 21 has been included to force the proposed algorithm to find the minimal tree.

path 31, 32, 22 has been selected. In this case, the overall half perimeter is 5 (10, 20, 30, 31, 32, 22). However, if we include the point 21 into the set $V$, the Algorithm 8.6 will be forced to find the optimal path (illustrated in Figure 8.16) with overall half perimeter equal to 4.

#### *8.3.8.4 Results*

Table 8.2 shows the average[2] wire length required to implement the TAMs. It shows the evaluated SoCs versus different number of available test pins. These wires are not required when NoC reuse is employed.

Table 8.2: Average wire length for dedicated TAMs. SoC set up to $c = 128$, $r =$*medium*, and $w = 32$.

| w | d695 | f2126 | g1023 | h953 | p22810 | p34392 | p93791 | q12710 | t512505 | u226 |
|---|---|---|---|---|---|---|---|---|---|---|
| 16 | 140 | 72 | 281 | 29 | 326 | 241 | 855 | 12 | 523 | 128 |
| 24 | 200 | 6 | 308 | 20 | 523 | 384 | 960 | 12 | 70 | 192 |
| 32 | 109 | 6 | 405 | 20 | 478 | 336 | 1106 | 12 | 119 | 256 |
| 40 | 233 | 6 | 226 | 20 | 508 | 341 | 838 | 12 | 95 | 254 |
| 48 | 275 | 6 | 399 | 20 | 708 | 113 | 564 | 12 | 95 | 176 |
| 56 | 314 | 6 | 509 | 20 | 386 | 376 | 1077 | 12 | 95 | 240 |
| 64 | 203 | 6 | 724 | 20 | 472 | 389 | 1960 | 12 | 96 | 240 |

Some systems like f2126, h953, and q12710 have bottleneck cores, i.e. cores that require a large percentage of the SoC test length to be tested. In these cases, a TAM usually has only one core (the bottleneck core). According to our estimation model, when there is only one core in a TAM, the required wire length is zero since we do not take into account the wires from the test pins to the CUT. For this reason these systems require fewer wires.

Table 8.3 shows the number of wires to implement a NoC with 32-bit width channels. The number of wires is presented for each system. For instance, system d695 has nine routers and 24 channels. Since each channel has 32 bits, then the NoC requires 24 × 32 = 768 wires[3].

[2] Considering ten different placements for each pair system-w.

[3] This wire count does not consider control wires used to implement protocol.

Table 8.3: Number of wires of a NoC with 32-bit width channels. Product of the number of channels of the NoC and the channel width.

| | d695 | f2126 | g1023 | h953 | p22810 | p34392 | p93791 | q12710 | t512505 | u226 |
|---|---|---|---|---|---|---|---|---|---|---|
| # routers | 9 | 4 | 16 | 9 | 25 | 16 | 36 | 4 | 36 | 9 |
| # channels | 24 | 8 | 48 | 24 | 80 | 48 | 120 | 8 | 120 | 24 |
| 32-NoC | 768 | 256 | 1536 | 768 | 2560 | 1536 | 3840 | 256 | 3840 | 768 |

Table 8.4 represents the relative number of wires to implement dedicated TAMs compared to the number of wires of the 32-bit NoC. For instance, the 140 wires required to implement dedicated TAMs for the d695 system considering 16 test wires (see Table 8.2 column d695 row 16) correspond to 18.23% (140/768), of wires of a 3x3 NoC considering channels of 32-bit width.

Table 8.4: Percentage of the wires of a NoC with 32-bit width channels.

| w | d695 | f2126 | g1023 | h953 | p22810 | p34392 | p93791 | q12710 | t512505 | u226 |
|---|---|---|---|---|---|---|---|---|---|---|
| 16 | 18.23 | 28.13 | 18.29 | 3.78 | 12.73 | 15.69 | 22.27 | 4.69 | 13.62 | 16.67 |
| 24 | 26.04 | 2.34 | 20.05 | 2.60 | 20.43 | 25.00 | 25.00 | 4.69 | 1.82 | 25.00 |
| 32 | 14.19 | 2.34 | 26.37 | 2.60 | 18.67 | 21.88 | 28.80 | 4.69 | 3.10 | 33.33 |
| 40 | 30.34 | 2.34 | 14.71 | 2.60 | 19.84 | 22.20 | 21.82 | 4.69 | 2.47 | 33.07 |
| 48 | 35.81 | 2.34 | 25.98 | 2.60 | 27.66 | 7.36 | 14.69 | 4.69 | 2.47 | 22.92 |
| 56 | 40.89 | 2.34 | 33.14 | 2.60 | 15.08 | 24.48 | 28.05 | 4.69 | 2.47 | 31.25 |
| 64 | 26.43 | 2.34 | 47.14 | 2.60 | 18.44 | 25.33 | 51.04 | 4.69 | 2.50 | 31.25 |

#### *8.3.8.5 Discussion*

The proposed measure does not count the wires required to connect the test pins and the wires for SoC-level test control logic because we assume that both test architectures based on dedicated TAMs and NoC reuse require these global wires.

The wire length required to implement a given TAM might also depend the order the cores are placed in the TAM. Let us assume a TAM involving the tiles 1, 2, and 3 of the system represented in Figure 8.15(b). If the test scheduling tool defines a TAM in the following order: { 1, 2, 3 }, then the number of hops is two. However, if the TAM is { 1, 3, 2 }, then the number of hops is three (two hops from 1 to 3 and one hop from 3 to 2). In this evaluation we assumed a core assignment order such that the number of hops is minimal. In the previous example we would assume { 1, 2, 3 } or { 3, 2, 1 }.

Our estimation algorithm gives the minimal Steiner tree for a given set of points. Thus, it returns the minimal number of wires to implement the TAM. On the other hand, in a actual chip, routing congestion might increase the actual wire length of the TAMs.

For these reasons, the results presented in Section 8.3.8.4 represent the minimal wire length to implement dedicated TAMs. More expensive wiring to implement dedicated TAMs is expected in actual designs.

## 8.4 Discussion

## 8.5 Limitations

Although the proposed test model is general, the implemented tool has some limitations such as:

1. *topology.*
   The current topology representation is based on matrix instead of on graph. It has been chosen to ease the implementation, but it imposes limitations on the supported topologies. For instance, just mesh topology is currently supported. Irregular topologies could be supported by changing the topology representation from matrix to graphs.

2. *routing algorithm.*
   The current results are based only on XY routing algorithm. A library of routing algorithms could be implemented such that the user could use one of the options in the library. The impact of routing algorithm shall be evaluated in the future.

## 8.6 Summary

This chapter presented a scheduling optimization tool for BE NoC-based systems that use the NoC to transport test data. The algorithm minimizes both test length and silicon area of the DfT logic. We compared the test length of the proposed approach with an algorithm used for test architectures based on conventional dedicated TAMs. The results show similar test length for both approaches. We think that the results presented in this chapter are a first step to actually define the advantages and drawbacks of NoC-reuse compared to dedicated TAMs. The second problem addressed in this chapter is the need for general NoC-reuse approaches. We have demonstrated that the proposed approach is more general than the previous one since it does not require a full cycle-accurate NoC model to determine the test scheduling. We have also proposed a model to evaluate the amount of wires required to implement dedicated TAMs. This model estimates the amount of wire length saving by using NoC to transport test data.

On the other hand, we reinforce that it is still required to test the proposed method with different NoCs features (different topologies, services, costs). In addition, we plan to compare the proposed approach with the packet-based and dedicated-path algorithms (COTA; LIU, 2006).

# 9 DFT OPTIMIZATION AND GENERATION

This chapter presents the procedure to determine the FIFO size for each vTAM. After test scheduling most information required to build an optimized test architecture is available except by the size of the FIFOs $d$ and the packet size $k$ (Figure 9.1). The procedure presented in Section 9.1 determines these information.

## 9.1 Proposed Algorithm for Buffer Sizing

The proposed optimization procedure presented in Algorithm 9.1 finds the minimal FIFO depth $d$ and packet size $k$ for each vTAM of the system such that $w$ test wires are sustained, reducing the DfT silicon area.

For a given vTAM with $w$ test wires and a given environment, like the one illustrated in Figure 9.2, the algorithm finds the minimal $d$ and $k$. The function `simul_env`$(d, k, w)$ runs a cycle-accurate simulation model of a partition, where this model can be configured in terms of $d$, $k$, and $w$. The simulation model consists of an ATE interface, the NoC, a dummy core with test wrapper, and a testbench emulating the ATE.

The model checks, during simulation, if $w$ test wires is supported without gaps. The algorithm initially sets the packet length to the maximal size supported by the network (line 1 in Algorithm 9.1). Next, the simulation model runs to test for gaps (line 3). In case gaps are found, the FIFO depth $d$ is incremented (line 5). Once the minimal $d$ is found, the next step iteratively finds the packet length in a linear search (line 8). The actual test data is not required for this simulation. Dummy data is used instead.

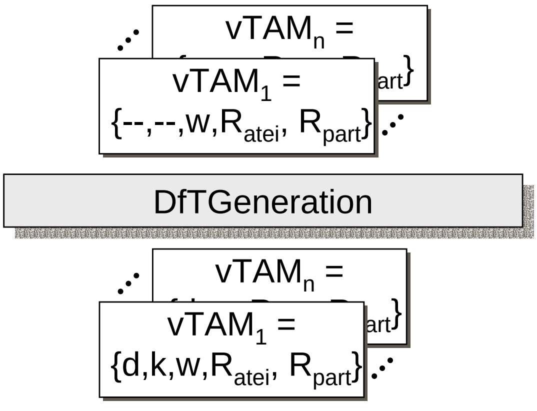

vTAM$_n$ =
part}
vTAM$_1$ =
{--,--,w,R$_{atei}$, R$_{part}$}
DfTGeneration
vTAM$_n$ =
art}
vTAM$_1$ =
{d,k,w,R$_{atei}$, R$_{part}$}

Figure 9.1: DfT generation step.

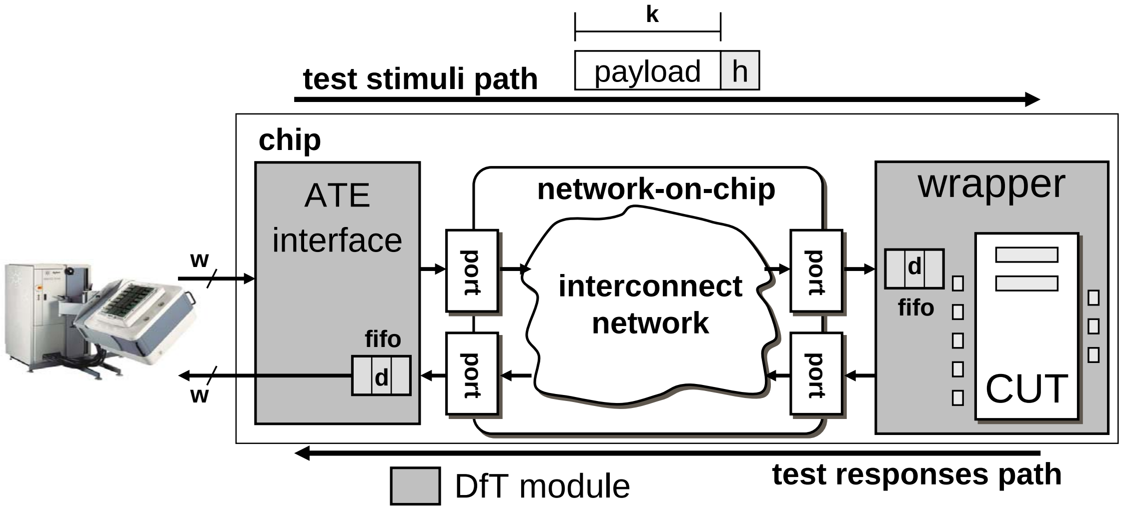


Figure 9.2: Conceptual simulation environment including the NoC, the partially generated DfT modules, and the parameters $d$, $k$, and $w$.

Algorithm 9.1: Minimize Buffer Size($w$)

```
gap := true; d := 0; k := ∞;
//search the FIFO depth
while(gap){
    gap := simul_env(d, k, w);
    if (gap) then
        d := d + 1;
}
k := 1; gap := true;
//search the packet length
while(gap){
    gap := simul_env(d, k, w);
    if (gap) then
        k := k + 1;
}
```

## 9.2 Results

Figure 9.3 illustrates two simulations runs of the simulation environment. They represent a simulation with and without gaps from the ATE to the wrapper. The NoC is a Hermes NoC (MORAES et al., 2004) with $c = 16$. The signal `fifo_words` represents the FIFO depth, `test_wires` represents $w$, `inputtestpins` represents the data sent from the ATE. The next two signals of the waveform represent data from the ATE interface to the first router. One can see that `mdatavalid` has periods of 4 clock cycles since $\left\lfloor \frac{c}{w} \right\rfloor = \left\lfloor \frac{16}{4} \right\rfloor$. The next four signals represent data from the router 00 to the 10 and from the router 10 to 11. One can see in signal `tx` that the periodicity is lost. The signal `sdataaccept` represents the data arriving at the wrapper input without periodicity. The last two signals represent the data after the FIFO. Figure 9.3(a) has gaps since the signal `read_en` is not periodic because the `fifo_words` is only one. On the other hand, Figure 9.3(a) has no gaps because the `fifo_words` is two, which in this case is sufficient to sustain four test wires. The simulation for finding the packet size $k$ is similar to this one for finding the FIFO depth $d$.

More results related to the proposed buffer sizing algorithm were presented in Section 5.4.3, page 97. The maximal FIFO depth found is five FIFO words, but it is required only when the maximal test bandwidth is required (i.e. $w = c/2$, the test bandwidth is half the physical bandwidth). In most cases smaller FIFOs are required.

## 9.3 Discussion

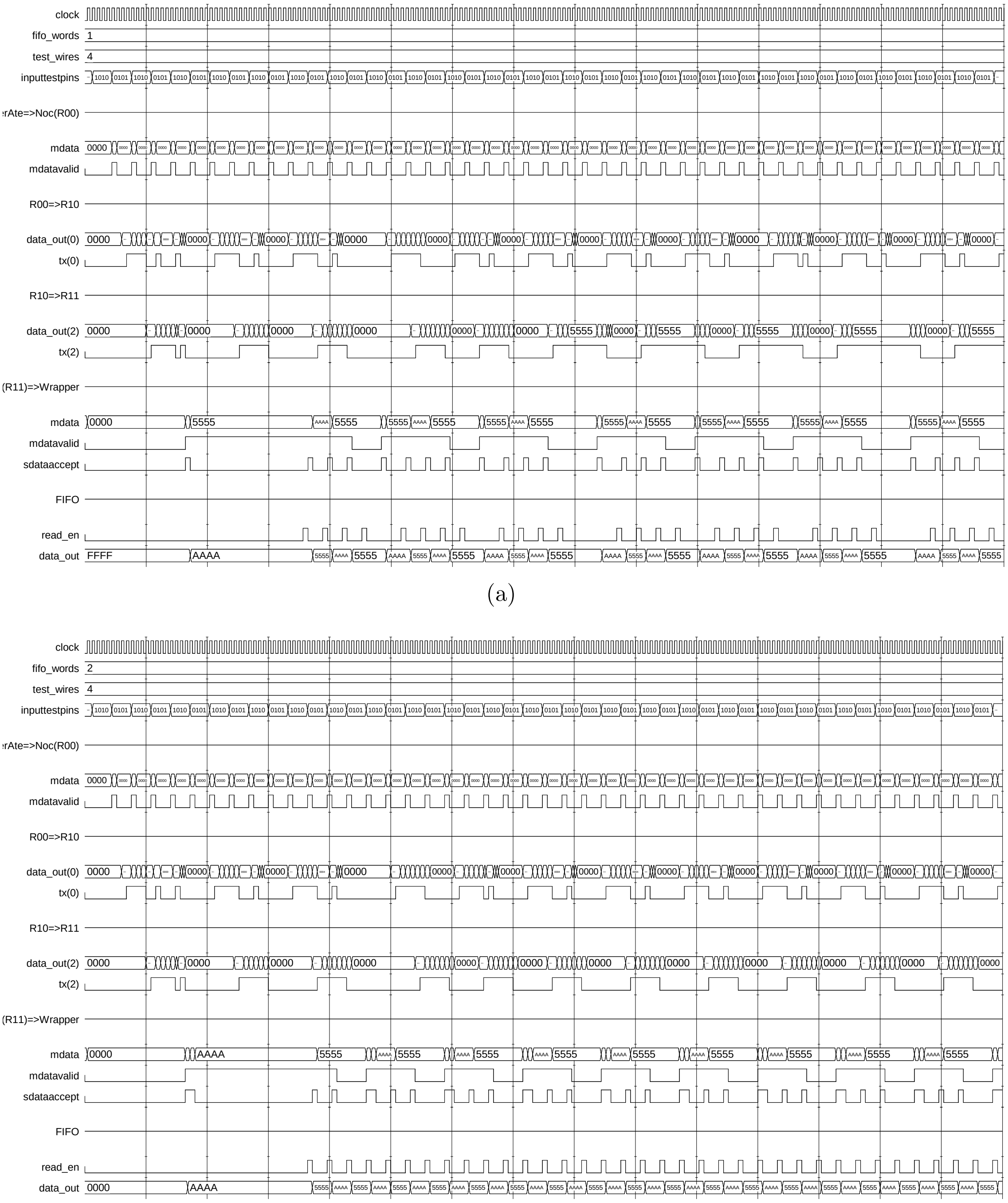
This algorithm relies on a cycle-accurate HDL model for sake of accuracy. For this reason the simulation of several parameters can be time consuming. On the other hand, it is executed only once per partition since all modules within a partition use the same $k$, $d$, and $w$ configuration.

## 9.4 Summary

This chapter proposed a simulation model to find the minimal FIFO depth and packet size for the NoC reuse approach. The proposed separation between test scheduling and the simulation model is crucial to define a general NoC reuse approach. On one hand, the test scheduling has no timing description of the NoC, in fact the NoC is seen as a set of pipelines driving constant and continuous data to the CUTs. On the other hand, all timing-dependent features of the proposed test approach are defined with the simulation environment presented in this chapter.

One can realize that the SoC test length is totally defined by the test scheduling without the need for this simulation step. The simulation is just used to define the minimal DfT silicon area. In addition, the proposed simulation environment can be easily created for a given NoC once the ATE interface and wrapper are configurable in terms of $k$, $d$, and $w$. It also relies on standard protocols to ease integration.

An analytical buffer sizing approaches shall be investigated in the future to enable faster DfT design.



(a)

(b)

Figure 9.3: Waveform of the simulation environment. Presence of gaps due to insufficient FIFO and packet sizes (a) and simulation without gaps (b). One can observe that there are no gaps in (b) because the signal `read_en` is periodic, i.e. it ticks every 4 clock cycles. This periodicity is sustained thanks to the FIFO of size two (see `fifo_words` in (b). The second signal top-down). On the other hand, if number of FIFO words is decrease to one, then it can be observed that the signal `read_en` in (a) (second signal bottom-up) is not periodic, causing gaps. Thus, this NoC requires FIFO of size two to sustain four test wires (see `test_wires`. The third signal.).

# 10 CONCLUSION

Currently, global interconnect solutions based on long wires, like buses, are being replaced by solutions based on shared and segmented wires, like NoCs, to reduce the cost of global interconnect. It has been demonstrated that conventional test-dedicated TAMs can also be considered long global wires, thus, TAMs are also subject to the same interconnect problems. For this reason, we have investigated the use of the NoC to transport test data during test application to avoid test-dedicated TAMs.

However, most of the research on NoC reuse has focused on high level test models for an specific case study. On one hand, this approach is important to introduce the NoC reuse problem and to demonstrate the viability of the approach. On the other hand, (*i*) accurate DfT models and (*ii*) generic test tools are required to enable the actual use of NoC reuse on existing NoC-based designs. Another important gap was (*iii*) the lack of quantitative comparisons between test architectures based on NoC reuse and on dedicated TAMs to actually define the advantage and drawbacks of both approaches. These three points are the main contributions of this thesis.

The rest of this chapter details the main contributions, the limitations, and future work.

## 10.1 Qualitative Analysis

### 10.1.1 Toward a General NoC-Reuse Approach

We believe that NoC reuse will be actually used if and only if general approaches that work for most NoC exist. We proposed a test model that can work for a large number of NoC instances. As far as we know this work is the first one to create awareness about the need for general NoC reuse approaches.

### 10.1.2 Compatibility with Conventional SoC Modular Testing

The advantage of similar and compatible solutions are numerous. For instance, a wrapper design considering hierarchical cores was proposed for conventional TAM. If a NoC reuse can work exactly like a dedicated TAM, then the chances that the same wrapper can be used for NoC reuse are high. The same conclusion can also be applied for algorithms, design tools, test equipments, etc. In conclusion, assuming similar assumptions reduce the required adaptation effort. It also makes a possible transition from the conventional to the proposed approach smoother.

Scheduling for NoC reuse had been seen as something completely different from scheduling with conventional TAM since the former involves NoCs which are much more complex and not standardized than conventional TAMs. Most authors advocate that new solutions are required. This thesis demonstrates the opposite. It just depends on how he problem is modeled.

For example, in terms of DfT, we show that a proper core terminal classification, adding protocol information and FIFO to the wrapper design is enough to adapt the conventional wrapper design to the NoC reuse environment.

Moreover, in terms of reused algorithms, we adapted the conventional algorithms for wrapper optimization with a very simple modification on the algorithm; by including the constraints of equal number of `SDI` and `SDO` terminals per test wire. In addition, we adapted a conventional algorithm for test scheduling by adding neighborhood and maximal TAM width constraints.

These examples show that the research done in the past based on conventional TAMs can be easily adapted for the NoC-reuse problem by using an appropriate underlying test model and compatible assumptions. We believe that our focus on compatibility might be fruitful in the near future because it will enable people to easily adapt previous methods for NoC reuse. Moreover it will ease a possible transition from dedicated TAMs to NoC as TAM.

### 10.1.3 Detailed DfT Design

As far as we know there were no papers focusing on DfT design for NoC reuse. The modification required in the test logic to enable NoC reuse were not known. Simplistic (i.e. do not consider important design issues) and unoptimized wrapper models were usually assumed in the test scheduling (see Section 5.5).

The results presented in (AMORY et al., 2006) were probably the first addressing the problem of DfT to enable NoC reuse, followed by (AMORY et al., 2007) and (AMORY et al., 2007). Actual HDL designs and DfT optimization tools were generated along this thesis. The HDL description of the DfT modules were simulated and, when it was possible, prototyped on FPGA for verification purposes.

For instance, we were the first to address the need for protocol conversion logic inside the wrapper to enable NoC reuse. This obligation has a significant impact on the increase of DfT silicon area. In addition, we have addressed the need for a new core test length expression. We have shown that the core test length for NoC reuse might be different compared to conventional wrapper.

In conclusion, by addressing practical problems in NoC-reuse we helped to define the required changes in the DfT logic to give support to NoC reuse.

### 10.1.4 Comparison with Conventional Test Architecture

As far as we know, we were the first to compare test length and silicon area between a NoC reuse approach and a conventional test approach based on dedicated TAM. This comparison demonstrates the actual advantages and drawbacks of both approaches.

## 10.2 Quantitative Analysis

Our quantitative analysis is based on test length and silicon area for the DfT logic.

### 10.2.1 SoC Test Length

Our results on test length show that the proposed test architecture is statistically as fast as the conventional architecture. On one hand, the neighborhood constraints increase the SoC test length. On the other hand, the core test length of the proposed approach is usually faster than the one for conventional wrapper. The combination of these two facts results in similar SoC test length. This result is surprising since we imagined that the proposed test approach would be slower due to the neighborhood and TAM width constraints.

### 10.2.2 Silicon Area for DfT Modules

We helped to define the actual advantages of NoC reuse compared to conventional TAMs in terms of required silicon area. For instance, most of prior work assumed very low or even no area overhead for DfT. We have shown that in fact the silicon area for DfT for NoC reuse is around 20% more expensive compared to DfT for dedicated TAMs. Most of prior work assumed that there would be no global long wires added into the chip just for test purposes. We have shown that most long wires were actually eliminated, however, some as the wires for test control logic and wires connecting the test pins are still required.

We proposed a new method to evaluate the wire length saving using NoC reuse. The results reported an average wire length saving from 2% to 51% compared to a NoC whose channel width has 32-bit.

## 10.3 Prospected Impact

We demonstrated the need for first establish actual DfT designs before defining high level test models and test scheduling tools. It is important to evaluate 'low-level' design issues like integration of DfT logic to the rest of the system, evaluate silicon area of the DfT, configurability of the DfT model, how easy it is to integrate the proposed logic to different NoCs, and characterize the DfT model in terms of test length. For instance, without working on wrapper design for NoC reuse we would not found the need for a different core test length formulation. We hope that this work motivates other researches on DfT for NoC reuse.

We created awareness of the need for compatible and general NoC reuse approach. Although we have focused on BE networks, we hope to motivate more research on general NoC reuse for other NoC domains (GT, BE, async).

## 10.4 Accomplished Goals and Contribution

These are the goals established in the Section 1.2 page 31 and how they were accomplished:

- *Make the requirements for NoC reuse explicit.*
  The problem statement presented in Section 4.7 defines all required variables for the test architecture optimization problem.

- *Determine the design of the required DfT modules.*
  Chapters 5.5, 6, and 7 presented the proposed DfT design and optimization procedures.

- *Determine the optimization algorithms for the DfT modules.*
  See last item.

- *Propose a test schedule tool for overall test architecture optimization.*
  The proposed design flow integrates the DfT optimization procedures with the test scheduling tool to generate an optimized test architecture.

- *Integrated framework for DfT modules, vTAMs, and SoC test length co-optimization.*
  The design flow and developed tools presented in Section 4.8, page 80, enables an overall reduction of silicon area for the DfT modules and SoC test length.

- *Implement a tool to evaluate wire length savings using NoC reuse.*
  This tool has been presented in Section 8.3.8, page 140.

- *Compare the proposed test architecture with the conventional test architecture based on dedicated TAM to establish the actual advantages and drawbacks of NoC reuse.*
  The results presented in Sections 8.3 and  5.4 support our comparison between both test approaches.

Some additional contributions include:

- Tools for wrapper optimization for both NoC reuse and dedicated TAMs;

- Tools for test scheduling for both NoC reuse and dedicated TAMs;

- Several small tools and scripts used to generate and gather data;

- HDL models for DfT modules.

## 10.5 Limitations and Issues Not Addressed

Although the following issues were not addressed in this thesis for lack of time, they should be investigated in the future:

- *Test scheduling tool*
  The test scheduling tool is not as general as the proposed test model. However, it is still more general than the approaches found so far; the reduced number of NoC information required demonstrates this fact. The test scheduling tool works for mesh and XY based NoCs without guaranteed services. With a little more programming effort the test scheduling tool will become as general as the proposed test model by supporting irregular topologies and multiple routing algorithms;

- *Wiring length from test pins to ATE interface*
  No procedure is proposed to minimize the wires required to connect the test pins to the DfT logic;

- *Wiring length of test control signals like scan enable, reset, and IEEE Std. 1500*
  No procedure is proposed to minimize the wires required for top-level test control logic;

- *Support other scheduling constraints like power consumption during test, precedence constraints, among others*
  It was out of the scope of this thesis to investigate other constraints rather than test length and silicon area for DfT since these features were not well established yet. However, these are relevant topics for future work;

- *Extend the model for NoCs with multiple frequencies and asynchronous NoCs*
  It was out of the scope of this thesis to investigate this kind of NoCs. However, these are also relevant topics for future work;

- *Comparison with other NoC reuse approach*
  It was out of the scope of this thesis to compare with other NoC reuse approach because there was no work comparing NoC reuse and conventional approach. Thus, the advantages and drawbacks of NoC reuse were not clear so far. Moreover, it is hard to accomplish such comparison due to the lack of benchmarks to evaluate NoC reuse approaches.

## 10.6 Future Work

This thesis has explored a wide range of problems related to NoC reuse as TAM. However, some of the proposed approaches can be improved (item 1 to 4) and more complete test models can be created (5 to 11) to support other design issues.

1. *Graph-based test scheduling tool to support topology independent NoCs*;

2. *Generate cost-effective wrapper for routers*;

3. *DfT for NoCs without identical routers*;

4. *Allow user to specify wrapper protocol*
   Currently only a subset of OCP is supported;

5. *Benchmarks for NoC reuse are required*
   Currently a fair comparison between the existent approaches is complex or not possible;

6. *Test scheduling for NoCs with guaranteed services*;

7. *Reusing asynchronous NoCs as TAM*;

8. *Interconnect test for NoC channels*;

9. *Testing systems with different test frequencies*;

10. *Support of hierarchical cores*;

11. *Testing Mixed-signal SoCs*;

12. *Evaluate other test constraints like power dissipation, thermal, and precedence.*

# REFERENCES

AITKEN, R. C. A Modular Wrapper Enabling High Speed BIST and Repair for Small Wide Memories. In: ITC, 2004. **Proceedings...** Washington: IEEE Computer Society, 2004. p.997–1005.

AKTOUF, C. A Complete Strategy for Testing an on-chip Multiprocessor Architecture. **IEEE Design & Test of Computers**, [S.l.], v.19, n.1, p.18–28, 2002.

ALLIANCE, V. **Virtual Component Interface Standard**. Available at: <http://www.vsi.org/>. Visited on: Oct. 2006.

AMORY, A. M.; BRIÃO, E. W.; COTA, E. F.; LUBASZEWSKI, M. S.; MORAES, F. G. A Scalable Test Strategy for Network-on-Chip Routers. In: ITC, 2005, Austin, Texas, USA. **Proceedings...** Los Alamitos: IEEE Computer Society, 2005. p.591–599.

AMORY, A. M.; COTA, E. F.; LUBASZEWSKI, M. S.; MORAES, F. G. Reducing Test Time with Processor Reuse in Network-on-Chip Based Systems. In: SBCCI, 2004, Porto de Galinhas, PE, Brazil. **Proceedings...** New York: ACM, 2004. p.111–116.

AMORY, A. M.; FERLINI, F.; LUBASZEWSKI, M. S.; MORAES, F. G. DfT for the Reuse of Networks-on-Chip as Test Access Mechanism. In: VTS, 2007, Berkeley, CA, USA. **Proceedings...** Los Alamitos: IEEE Computer Society, 2007. p.435–440.

AMORY, A. M.; GOOSSENS, K.; MARINISSEN, E. J.; LUBASZEWSKI, M. S.; MORAES, F. G. Wrapper Design for the Reuse of Networks-on-Chip as Test Access Mechanism. In: ETS, 2006, Southampton, UK. **Proceedings...** Los Alamitos: IEEE Computer Society, 2006. p.213–218.

AMORY, A. M.; GOOSSENS, K.; MARINISSEN, E. J.; LUBASZEWSKI, M. S.; MORAES, F. G. Wrapper Design for the Reuse of a Network on Chip or Other Functional Interconnect as Test Access Mechanism. **IET Computers & Digital Techniques**, [S.l.], v.1, n.3, p.197–206, 2007.

AMORY, A. M.; LUBASZEWSKI, M. S.; MORAES, F. G. A Programmable Logic BIST Controller for IP Cores. In: LATW, 2004, Cartagena de Indias, Colombia. **Proceedings...** [S.l.: s.n.], 2004. p.104–109.

AMORY, A. M.; LUBASZEWSKI, M. S.; MORAES, F. G.; MORENO, E. I. Test Time Reduction Reusing Multiple Processors in a Network-on-Chip Based Architecture. In: DATE, 2005, Munich, Germany. **Proceedings...** Washington: IEEE Computer Society, 2005. v.1, p.62–63.

AMORY, A. M.; OLIVEIRA, L. A.; MORAES, F. G. Software-Based Test for Non-Programmable Cores in Bus-Based System-on-Chip Architectures. In: VLSI-SOC, 2003, Darmstadt, Germany. **Proceedings...** [S.l.: s.n.], 2003. p.174–179.

APPELLO, D. et al. Exploiting Programmable BIST for the Diagnosis of Embedded Memory Cores. In: ITC, 2003. **Proceedings...** [S.l.: s.n.], 2003. p.379–385.

ARABI, K. Logic BIST and Scan Test Techniques for Multiple Identical Blocks. In: VTS, 2002, Monterey, CA, USA. **Proceedings...** Los Alamitos: IEEE Computer Society, 2002. p.60–68.

ARM. **AMBA AXI Protocol Specification. Version 1.0**. Available at: <http://www.arm.com/products/solutions/AMBA3AXI.html>. Visited on: Oct. 2006.

ARM. **AMBA 3 APB Protocol Specification. Version 1.0**. Available at: <http://www.arm.com/products/solutions/AMBAAPB.html>. Visited on: Oct. 2006.

ARTERIS. **Arteris Introduces Industry's First Products for Building Networks on Chip (NoC)**. Available at: <http://www.us.design-reuse.com/news/news9888.html>. Visited on: Mar. 2006.

ARTERIS. **Arteris Web Site**. Available at: <http://www.arteris.net/>. Visited on: Mar. 2006.

BAINBRIDGE, J.; FURBER, S. CHAIN: A Delay Insensitive CHip Area INterconnect. **IEEE Micro**, [S.l.], v.22, n.5, p.16–23, 2002.

BARDELL, P. H. **Built-In Test for VLSI, Pseudorandom Techniques**. [S.l.]: John Wiley & Sons, 1987.

BEEST, F. T. et al. Automatic Scan Insertion and Test Generation for Ssynchronous Circuits. In: ITC, 2002. **Proceedings...** [S.l.: s.n.], 2002. p.804–813.

BENINI, L.; DE MICHELI, G. Networks on Chips: A New SoC Paradigm. **IEEE Computer**, [S.l.], v.35, n.1, p.70–80, 2002.

BENINI, L.; DE MICHELI, G. **Networks on Chips, Technology and Tools**. [S.l.]: Morgan Kaufmann Publishers, 2006.

BHUNIA, S. et al. A Novel Low-Overhead Delay Testing Technique for Arbitrary Two-Pattern Test Application. In: DATE, 2005. **Proceedings...** Washington: IEEE Computer Society, 2005. p.1136–1141.

BJERREGAARD, T. **The MANGO Clockless Network-on-Chip**: Concepts and Implementation. 2005. Tese (Doutorado em Ciência da Computação) — Technical University of Denmark.

BJERREGAARD, T.; MAHADEVAN, S. A Survey of Research and Practices on Network-on-Chip. **ACM Computing Surveys**, [S.l.], v.38, n.1, 2006.

BJERREGAARD, T.; SPARSO, J. A Router Architecture for Connection-Oriented Service Guarantees in the MANGO Clockless Network-on-Chip. In: DATE, 2005. **Proceedings...** Washington: IEEE Computer Society, 2005. p.1226–1231.

BJERREGAARD, T.; SPARSO, J. Scheduling Discipline for Latency and Bandwidth Guarantees in Asynchronous Network-on-Chip. In: ASYNC, 2005. **Proceedings...** [S.l.: s.n.], 2005. p.34–43.

BURDASS, A. et al. Embedded Test and Debug of Full Custom and Synthesisable Microprocessor Cores. In: ETW, 2000. **Proceedings...** [S.l.: s.n.], 2000. p.17–22.

BUSHNELL, M. L.; AGRAWAL, V. D. **Essentials of Electronic Testing for Digital, Memory, and Mixed-Signal VLSI Circuits**. [S.l.]: Kluwer Academic Publishers, 2000.

CAST. **C1394A IEEE-1394a Link Layer Controller Core**. Available at: <http://www.cast-inc.com/cores/c1394a/index.shtml>. Visited on: Oct. 2006.

CAST. **PCIe-EP PCI Express Endpoint Controller Core**. Available at: <http://www.cast-inc.com/cores/pcie-ep/index.shtml>. Visited on: Oct. 2006.

CAST. **GPIO8 General Purpose Input/Output Unit Core**. Available at: <http://www.cast-inc.com/cores/gpio8/index.shtml>. Visited on: Oct. 2006.

CHAKRABARTY, K. Design of System-on-a-Chip Test Access Architectures Using Integer Linear Programming. In: VTS, 2000. **Proceedings...** Los Alamitos: IEEE Computer Society, 2000. p.127–134.

CIORDA, C. et al. NoC Monitoring: Impact on the Design Flow. In: ISCAS, 2006. **Proceedings...** [S.l.: s.n.], 2006. p.1981–1984.

CLERMIDY, F.; VARREAU, D.; LATTARD, D. **A NoC-Based Communication Framework for Seamless IP Integration in Complex Systems**. Available at: <http://www.us.design-reuse.com/articles/article12226.html>. Visited on: Mar. 2006.

COTA, E. F.; BRISOLARA, L.; CARRO, L.; SUSIN, A.; LUBASZEWSKI, M. S. MET: A Microprocessor for Embedded Test. In: IEEE TEST OF CORE-BARED SYSTEMS, 2001. **Proceedings...** [S.l.: s.n.], 2001. p.100–107.

COTA, E. F.; CARRO, L.; LUBASZEWSKI, M. S. Reusing an On-Chip Network for the Test of Core-based Systems. **ACM TODAES**, New York, NY, USA, v.9, n.4, p.471–499, 2004.

COTA, E. F.; CARRO, L.; LUBASZEWSKI, M. S.; ORAILOGLU, A. Test Planning and Design Space Exploration in a Core-Based Environment. In: DATE, 2002, Paris, France. **Proceedings...** Washington: IEEE Computer Society, 2002. p.478–485.

COTA, E. F.; KREUTZ, M.; ZEFERINO, C. A.; CARRO, L.; LUBASZEWSKI, M. S.; SUSIN, A. The Impact of NoC Reuse on the Testing of Core-based Systems. In: VTS, 2003, Napa Valley, CA, USA. **Proceedings...** Washington: IEEE Computer Society, 2003. p.128–133.

COTA, E. F.; LIU, C. Constraint-Driven Test Scheduling for NoC-Based System. **IEEE Trans. on CAD of Integrated Circuits and Systems**, Sonoma, CA, USA, v.25, n.11, p.2465–2478, 2006.

DALLY, W. J.; TOWLES, B. Route Packets, not Wires: on-Chip Interconnection Networks. In: DAC, 2001. **Proceedings...** [S.l.: s.n.], 2001. p.684–689.

DUATO, J.; YALAMANCHILI, S.; NI, L. **Interconnection Networks**: An Engineering Approach. [S.l.]: Morgan Kaufmann Publishers, 2003.

EFTHYMIOU, A.; BAINBRIDGE, J.; EDWARDS, D. Adding Testability to an Asynchronous Interconnect for GALS SoC. In: ATS, 2004. **Proceedings...** [S.l.: s.n.], 2004. p.20–23.

EFTHYMIOU, A.; BAINBRIDGE, J.; EDWARDS, D. Test Pattern Generation and Partial-Scan Methodology for an Asynchronous SoC Interconnect. **IEEE Trans. on VLSI Systems**, [S.l.], v.13, n.12, p.1384–1393, 2005.

FAGOT, C. et al. On Calculating Efficient LFSR Seeds for Built-In Self-Test. In: ETW, 1999. **Proceedings...** [S.l.: s.n.], 1999. p.7–14.

FEIGE, C. et al. Integration of the Scan-Test Method into an Architecture Specific Core-Test Approach. In: ETW, 1998. **Proceedings...** [S.l.: s.n.], 1998. p.241–245.

FELICIJAN, T.; FURBER, S. An Asynchronous On-Chip Network Router with Quality-of-Service (QoS) Support. In: INTERNATIONAL SOC CONFERENCE, 2004. **Proceedings...** [S.l.: s.n.], 2004. p.274–277.

GAISLER. **Leon2 Core**. Available at: <http://www.gaisler.com/>. Visited on: Jan. 2006.

GOEL, S. K.; MARINISSEN, E. J. SOC Test Architecture Design for Efficient Utilization of Test Bandwidth. **ACM TODAES**, New York, NY, USA, v.8, n.4, p.399–429, 2003.

GOEL, S. K.; MARINISSEN, E. J. Layout-Driven SOC Test Architecture Design for Test Time and Wire Length Minimization. In: DATE, 2003. **Proceedings...** Washington: IEEE Computer Society, 2003. p.738–743.

GOOR, A. J. van de; SCHANSTRA, I.; ZORIAN, Y. Functional Test for Shifting-Type FIFOs. In: EDTC, 1995. **Proceedings...** [S.l.: s.n.], 1995. p.133–138.

GOOSSENS, K.; DIELISSEN, J.; RADULESCU, A. The Æthereal Network on Chip: Concepts, Architectures, and Implementations. **IEEE Design & Test of Computers**, [S.l.], v.22, n.5, p.21–31, 2005.

GOOSSENS, K. et al. Networks on Silicon: Combining Best-Effort and Guaranteed Services. In: DATE, 2002. **Proceedings...** Washington: IEEE Computer Society, 2002. p.423–425.

GOOSSENS, K. et al. A Design Flow for Application-Specific Networks on Chip with Guaranteed Performance to Accelerate SOC Design and Verification. In: DATE, 2005. **Proceedings...** Washington: IEEE Computer Society, 2005. p.1182–1187.

GRECU, C. et al. Methodologies and Algorithms for Testing Switch-Based NoC Interconnects. In: DFT, 2005. **Proceedings...** [S.l.: s.n.], 2005. p.238–246.

GUERRIER, P.; GREINER, A. A Generic Architecture for On-Chip Packet-Switched Interconnections. In: DATE, 2000. **Proceedings...** New York: ACM, 2000. p.250–256.

HARROD, P. Testing Reusable IP - A Case Study. In: ITC, 1999. **Proceedings...** [S.l.: s.n.], 1999. p.493–498.

HELLEBRAND, S. et al. Buil-In Test for Circuits with Scan Based on Reseeding of Multiple-Polynomial Linear Feedback Shift Resgisters. **IEEE Trans. on Computers**, [S.l.], v.44, n.2, p.223–233, 1995.

HELLEBRAND, S.; WUNDERLICH, H. J.; HERTWIG, A. Mixed-Mode BIST Using Embedded Processors. In: ITC, 1996. **Proceedings...** [S.l.: s.n.], 1996. p.195–204.

HENKEL, J. Closing the SoC Design Gap. **Computer**, [S.l.], v.36, n.6, p.119–121, 2003.

HETHERINGTON, G. et al. Logic BIST for Large Industrial Designs: Real Issues and Case Studies. In: ITC, 1999. **Proceedings...** [S.l.: s.n.], 1999. p.358–367.

HOSSEINABADY, M. et al. Concurrent Testing of Switches in NoC-Based SoCs. In: DATE, 2006. **Proceedings...** Leuven: European Design and Automation Association, 2006. p.1–6.

HUANG, C.-T. et al. A Programmable BIST Core for Embedded DRAM. **IEEE Design & Test of Computers**, [S.l.], v.16, n.1, p.59–70, 1999.

HUANG, J.-R. et al. A Self-Test Methodology for IP Cores in Bus-Based Programmable SoCs. In: VTS, 2001. **Proceedings...** Los Alamitos: IEEE Computer Society, 2001. p.198–203.

HUANG, Y. et al. Resource Allocation and Test Scheduling for Concurrent Test of Core-Based SOC Design. In: ATS, 2001. **Proceedings...** [S.l.: s.n.], 2001. p.265–270.

HWANG, S.; ABRAHAM, J. A. Reuse of Adrdressable System Bus for SOC Testing. In: IEEE ASIC/SOC, 2001. **Proceedings...** [S.l.: s.n.], 2001. p.215–219.

HWANG, S.; ABRAHAM, J. A. Test Data Compression and Test Time Reduction Using an Embedded Microprocessor. **IEEE Trans. on VLSI Systems**, [S.l.], v.11, n.5, p.853–862, 2003.

IBM. **CoreConnect Bus Architecture. Version 2.2**. Available at: <http://www-03.ibm.com/chips/products/coreconnect/>. Visited on: Oct. 2006.

IEEE. **IEEE Std 1394c-2006**. Available at: <http://grouper.ieee.org/groups/1394/c/>. Visited on: Oct. 2006.

ITRS. **International Technology Roadmap for Semiconductors**. Available at: <http://www.itrs.net/>. Visited on: Jan. 2007.

IYENGAR, V.; CHAKRABARTY, K. Precedence-Based, Preemptive, and Power-Constrained Test Scheduling for System-on-a-Chip. In: VTS, 2001. **Proceedings...** Los Alamitos: IEEE Computer Society, 2001. p.368–374.

IYENGAR, V.; CHAKRABARTY, K.; MARINISSEN, E. J. Co-Optimization of Test Wrapper and Test Access Architecture for Embedded Cores. **Journal of Electronic Testing: Theory and Applications**, [S.l.], v.18, n.2, p.213–230, 2002.

IYENGAR, V.; CHAKRABARTY, K.; MARINISSEN, E. J. On Using Rectangle Packing for SOC Wrapper/TAM Co-Optimization. In: VTS, 2002. **Proceedings...** Los Alamitos: IEEE Computer Society, 2002. p.253–258.

JANTSCH, A.; TENHUNEN, H. **Networks on Chip**. [S.l.]: Kluwer Academic Publishers, 2003.

JUTMAN, A. At-Speed On-Chip Diagnosis of Board-Level Interconnect Faults. In: ETS, 2004. **Proceedings...** [S.l.: s.n.], 2004. p.2–7.

JUTMAN, A.; UBAR, R.; RAIK, J. New Built-In Self-Test Scheme for SoC Interconnect. In: DESIGN AND DIAGNOSTIC OF ELECRONIC CIRCUITS AND SYSTEMS, 2005. **Proceedings...** [S.l.: s.n.], 2005. p.224–227.

KAHNG, A.; ROBINS, G. A New Class of Steiner Tree Heuristics with Good Performance. **IEEE Trans. on CAD of Integrated Circuits and Systems**, [S.l.], v.11, n.7, p.893–902, 1992.

KASTENSMIDT, F.; COTA, E. F.; CASSEL, M.; MEIRELLES, P.; AMORY, A. M.; LUBASZEWSKI, M. S. Redefining and Testing Interconnect Faults in Mesh NoCs. In: ITC, 2007, Santa Clara, CA, USA. **Proceedings...** [S.l.: s.n.], 2007.

KEUTZER, K. et al. System-Level Design: Orthogonalization of Concerns and Platform-Based Design. **IEEE Trans. on CAD of Integrated Circuits and Systems**, [S.l.], v.19, n.12, p.1523–1543, 2000.

KORANNE, S. A Novel Reconfigurable Wrapper for Testing of Embedded Core-Based SOCs and its Associated Scheduling Algorithm. **Journal of Electronic Testing: Theory and Applications**, [S.l.], v.18, n.4/5, p.415–434, Aug. 2002.

KORANNE, S. Formulating SoC Test Scheduling as a Network Transportation Problem. **IEEE Trans. on CAD of Integrated Circuits and Systems**, [S.l.], v.21, n.12, p.1517–1525, Dec. 2002.

KRISHNA, C. V.; JAS, A.; TOUBA, N. A. Test Vector Encoding Using Partial LFSR Reseeding. In: ITC, 2001. **Proceedings...** [S.l.: s.n.], 2001. p.885–893.

KRISHNA, C. V.; TOUBA, N. A. Reducing Test Data Volume Using LFSR Reseeding with Seed Compression. In: ITC, 2002. **Proceedings...** [S.l.: s.n.], 2002. p.321–330.

KRSTIC, A. et al. Embedded Software-Based Self-Test for Programmable Core-Based Designs. **IEEE Design & Test of Computers**, [S.l.], v.19, n.4, p.18–27, 2002.

LAI, W.-C.; CHENG, K.-T. Instruction-Level DFT for Testing Processor and IP Cores in System-on-a-Chip. In: DAC, 2001. **Proceedings. . .** [S.l.: s.n.], 2001. p.59–64.

LEE S-J. LEE, K.; YOO, H.-J. Analysis and Implementation of Practical, Cost-Effective Networks on Chips. **IEEE Design & Test of Computers**, [S.l.], v.22, n.5, p.422–433, 2005.

LIU, C.; IYENGAR, V.; SHI, J.; COTA, E. F. Power-Aware Test Scheduling in Network-on-Chip Using Variable-Rate On-Chip Clocking. In: VTS, 2005, Palm Springs, CA, USA. **Proceedings. . .** Los Alamitos: IEEE Computer Society, 2005. p.349–354.

MAGARSHACK, P.; PAULIN, P. G. System-on-Chip Beyond the Nanometer Wall. In: DAC, 2003. **Proceedings. . .** [S.l.: s.n.], 2003. p.419–424.

MARCON, C. A. M.; AMORY, A. M.; LUBASZEWSKI, M. S.; SUSIN, A.; CALAZANS, N.; MORAES, F. G. Applying Memory Test Algorithms to Embedded Systems. In: LATW, 2004, Cartagena de Indias, Colombia. **Proceedings. . .** [S.l.: s.n.], 2004. p.43–48.

MARINISSEN, E. J. et al. A Structured and Scalable Mechanism for Test Access to Embedded Reusable Cores. In: ITC, 1998. **Proceedings. . .** [S.l.: s.n.], 1998. p.284–293.

MARINISSEN, E. J. et al. On IEEE P1500's Standard for Embedded Core Test. **Journal of Electronic Testing: Theory and Applications**, [S.l.], v.18, p.365–383, 2002.

MARINISSEN, E. J.; GOEL, S. K.; LOUSBERG, M. Wrapper Design for Embedded Core Test. In: ITC, 2000. **Proceedings. . .** [S.l.: s.n.], 2000. p.911–920.

MARINISSEN, E. J.; IYENGAR, V.; CHAKRABARTY, K. A Set of Benchmarks for Modular Testing of SOCs. In: ITC, 2002. **Proceedings. . .** [S.l.: s.n.], 2002. p.519–528.

MARINISSEN, E. J.; LOUSBERG, M. The Role of Test Protocols in Testing Embedded-Core-Based System ICs. In: ETW, 1999. **Proceedings. . .** [S.l.: s.n.], 1999. p.70–75.

MCCABE, J. D. **Network Analysis, Architecture, and Design**. 2nd ed. [S.l.]: Morgan Kaufmann Publishers, 2003.

MENTOR GRAPHICS. **Mentor Graphics Web Site**. Available at: <http://www.mentor.com//>. Visited on: Apr. 2007.

MILLBERG, M. et al. Guaranteed Bandwidth using Looped Containers in Temporally Disjoint Networks within the Nostrum Network on Chip. In: DATE, 2004. **Proceedings. . .** Washington: IEEE Computer Society, 2004. v.2, p.890–895.

MORAES, F. G. et al. HERMES: an Infrastructure for Low Area Overhead Packet-switching Networks on Chip. **Integration, the VLSI Journal**, [S.l.], p.69–93, 2004.

MURALI, S. et al. A Methodology for Mapping Multiple Use-Cases on to Networks on Chip. In: DATE, 2006. **Proceedings...** Leuven: Belgium: European Design and Automation Association, 2006.

OCP-IP. **Open Core Protocol Specification. Release 2.1**. Available at: <http://www.ocpip.org/>. Visited on: Oct. 2006.

OPENCORES. **WISHBONE System-on-Chip Interconnection Architecture for Portable IP Cores. Revision B.3**. Available at: <http://www.opencores.org/projects.cgi/web/wishbone/wishbone>. Visited on: Oct. 2006.

OPENCORES. **Plasma Processor**. Available at: <http://www.opencores.org/>. Visited on: Jan. 2006.

OST, L. et al. MAIA - A Framework for Networks on Chip Generation and Verification. In: ASPDAC, 2005. **Proceedings...** [S.l.: s.n.], 2005. p.49–52.

PCI-SIG. **PCI Express Base Specification 1.1**. Available at: <http://www.pcisig.com/specifications/pciexpress/>. Visited on: Oct. 2006.

PHILIPS SEMICONDUCTORS. **Device Transaction Level Protocol Specification. Version 2.2**. 2002.

PREAS, B. T.; LORENZETTI, M. J. **Physical Design Automation of VLSI Systems**. [S.l.]: Benjamin Cummings Publishing Company, 1988.

RADULESCU, A. et al. An Efficient On-Chip Network Interface Offering Guaranteed Services, Shared-Memory Abstraction, and Flexible Network Programming. **IEEE Trans. on CAD of Integrated Circuits and Systems**, [S.l.], v.24, n.1, p.4–17, 2005.

RAJSKI, J. et al. Embedded Deterministic Test for Low-Cost Manufacturing. **IEEE Design & Test of Computers**, [S.l.], v.20, n.5, p.58–66, 2003.

RAJSKI, J.; TYSZER, J.; ZACHARIA, N. Test Data Decompression for Multiple Scan Designs with Boundary Scan. **IEEE Trans. on Computers**, [S.l.], v.47, n.11, p.1188–1200, 1998.

RAMBUS. **Rambus FlexIO**. Available at: <http://www.rambus.com/>. Visited on: Oct. 2006.

RAMBUS. **Rambus XDR**. Available at: <http://www.rambus.com/>. Visited on: Oct. 2006.

RAMBUS. **Rambus RDRAM**. Available at: <http://www.rambus.com/>. Visited on: Oct. 2006.

RAPIDIO. **RapidIO Specifications**. Available at: <http://www.rapidio.org/specs/current>. Visited on: Oct. 2006.

REARICK, J. Practical Scan Test Generation and Application for Embedded FIFOs. In: ITC, 1999. **Proceedings. . .** [S.l.: s.n.], 1999. p.294–300.

RIJPKEMA, E. et al. Trade Offs in the Design of a Router with Both Guaranteed and Best-Effort Services for Networks on Chip. In: DATE, 2003. **Proceedings. . .** Washington: IEEE Computer Society, 2003. p.350–355.

SAASTAMOINEN, I.; ALHO, M.; NURMI, J. Buffer Implementation for Proteo Network-on-Chip. In: ISCAS, 2003. **Proceedings. . .** [S.l.: s.n.], 2003. v.2, p.113–116.

SALTZER, J. H.; REED, D. P.; CLARK, D. D. End-to-End Arguments in System Design. **ACM Trans. in Computer Systems**, [S.l.], v.2, n.4, p.277–288, 1984.

SILISTIX. **Silistix Web Site**. Available at: <http://www.silistix.com//>. Visited on: Mar. 2006.

SILVA, F. da (Ed.). **IEEE Std 1500$^{TM}$-2005**: IEEE Standard Testability Method for Embedded Core-based Integrated Circuits. New York: IEEE, 2005.

SILVA, F. da; MCLAURIN, T.; WAAYERS, T. **The Core Test Wrapper Handbook**: Rationale and Application of IEEE Std. 1500$^{TM}$. [S.l.]: Springer, 2006.

STMICROELECTRONICS. **STNoC, Building a New System-on-Chip Paradigm**. Available at: <http://www.st.com/stonline/press/news/back2005/b9014t.htm>. Visited on: Mar. 2006.

STROUD, C. E. **A Designer's Guide to Built-In Self-Test**. [S.l.]: Kluwer Academic, 2002.

TANENBAUM, A. S. **Computer Networks**. 3rd ed. [S.l.]: Prentice-Hall, 1996.

TRAN, X.-T. et al. A DFT Architecture for Asynchronous Networks-on-Chip. In: ETS, 2006. **Proceedings. . .** [S.l.: s.n.], 2006. p.219–224.

UBAR, R.; RAIK, J. Testing Strategies for Network on Chip. In: JANTSCH, A.; TENHUNEN, H. (Ed.). **Networks on Chip**. [S.l.]: Kluwer Academic Publisher, 2003. p.131–152.

USB. **USB 2.0 Specification**. Available at: <http://www.usb.org/developers/docs/>. Visited on: Oct. 2006.

VARMA, P.; BHATIA, S. A Structured Test Re-Use Methodology for Core-Based System Chips. In: ITC, 1998. **Proceedings. . .** [S.l.: s.n.], 1998. p.294–302.

VERMEULEN, B. et al. Bringing Communication Networks On Chip: Test and Verification Implications. **IEEE Communications Magazine**, [S.l.], v.41, n.9, p.74–81, 2003.

WANG, X. et al. Asynchronous Network Node Design for Network-on-Chip. In: INTERNATIONAL SYMPOSIUM ON SIGNALS, CIRCUITS AND SYSTEMS, 2005. **Proceedings. . .** [S.l.: s.n.], 2005. p.55–58.

WEBER, W.-D. et al. A Quality-of-Service Mechanism for Interconnection Networks in System-on-Chips. In: DATE, 2005. **Proceedings...** Washington: IEEE Computer Society, 2005. p.1232–1237.

WU, Y.; MACDONALD, P. Testing ASICs with Multiple Identical Cores. **IEEE Trans. on CAD of Integrated Circuits and Systems**, [S.l.], v.22, n.3, p.327–336, 2003.

XU, Q.; NICOLICI, N. Resource-Constrained System-on-a-Chip Test: a Survey. **IEE Computers & Digital Techniques**, [S.l.], v.152, n.1, p.67–81, 2005.

ZEFERINO, C. A.; SUSIN, A. A. SoCIN: A Parametric and Scalable Network-on-Chip. In: SBCCI, 2003. **Proceedings...** [S.l.: s.n.], 2003. p.121–126.

ZHANG, H. Service Disciplines for Guaranteed Performance Service in Packet-Switching Networks. **Proc. of the IEEE**, [S.l.], v.83, n.10, p.1374–1396, 1995.

ZORIAN, Y.; MARINISSEN, E. J.; DEY, S. Testing Embedded-Core Based System Chips. In: ITC, 1998. **Proceedings...** [S.l.: s.n.], 1998. p.130–143.

# APPENDIX A

## Lógica e Escalonamento de Teste para Sistemas com Redes Intra-Chip Baseadas em Topologia de Malha

Esta seção apresenta os objetivos e principais contribuições desta tese.

Com a crescente complexidade dos sistemas computacionais, em especial sistemas em um único substrato (SoC - System-on-Chip), há a necessidade de utilizar técnicas de reuso para atingir as metas de time-to-market. Apesar das vantagens inerentes ao reuso de núcleos, identifica-se quatro grandes problemas que devem ser resolvidos para que se possa construir facilmente um SoC: (i) como integrar núcleos entre si; (ii) quais linguagens para descrição de sistemas usar; (iii) como proteger a propriedade intelectual do autor e do usuário do núcleo; (iv) como testar projetos baseados em núcleos. Destes quatro problemas, este trabalho se limita à integração de núcleos e ao teste do sistema. A forma usual de integrar os núcleos entre si é através de barramentos. A interconexão por barramento é simples, do ponto de vista de implementação, apresentando entretanto diversas desvantagens (BENINI; DE MICHELI, 2002): (i) apenas uma troca de dados pode ser realizada por vez, pois o meio físico é compartilhado por todos os núcleos, reduzindo o desempenho global do sistema; (ii) há a necessidade de mecanismos inteligentes de arbitragem do meio físico para evitar desperdício de largura de banda; (iii) a escalabilidade é limitada, ou seja, o número de núcleos que podem ser conectados ao barramento é muito baixo, tipicamente da ordem de dezenas; (iv) o uso de linhas globais em um CI (Circuito Integrado) com tecnologia submicrônica impõe sérias restrições ao desempenho do sistema devido aos elementos parasitas inerentes a fios longos. Como alternativa a barramentos, trabalhos recentes (BENINI; DE MICHELI, 2002; ZEFERINO; SUSIN, 2003; BJERREGAARD; MAHADEVAN, 2006; MORAES et al., 2004) utilizam redes de interconexão para conectar núcleos de hardware em um SoC. Este tipo de conexão, chamada de NoC (em inglês, Network-on-Chip), atribui ao SoC características de reuso e escalabilidade em relação ao número de núcleos que podem ser integrados no sistema.

Uma NoC é formada por roteadores e canais. Todos os núcleos do sistema são conectados à NoC. Outra dificuldade relacionada ao projeto de SoCs é seu teste. Com a grande integração dos SoCs, há uma grande quantidade de dados de teste que necessitam ser transferidos do testador para o circuito. O estado da arte em testadores deve ser utilizado para alcançar as restrições tecnológicas do circuito. Porém, estes testadores possuem alto custo devido à grande necessidade de memória, largura de banda, número de pinos e velocidade limitada. Uma das principais motivações da pesquisa de teste de sistemas integrados é manter estes custos de teste em níveis aceitáveis de forma a viabilizar a sua comercialização.

O uso de NoCs pode trazer vantagens em relação a teste. O método de teste tradicional de SoCs compreende a criação de caminhos dos pinos aos núcleos que são usados exclusivamente durante o teste para o transporte de estímulos e respostas de teste. Quando se usa NoCs para integrar os núcleos, estes caminhos não somente já estão disponíveis como também é possível ter vários destes caminhos ativos em paralelo. O reuso da NoC para teste é interessante pois viabiliza o teste de um maior número de núcleos em paralelo, oque reduz o tempo de teste do sistema, sem acréscimo de área. Sistemas baseados em NoCs disponibilizam naturalmente múltiplos caminhos dos pinos aos núcleos. Entretanto, o tempo de teste pode ser sub-ótimo se o número de pinos conectados ao testador externo não for suficiente para usar toda a largura de banda disponibilizada pela NoC. Acrescentar pinos exclusivamente para o teste é uma solução cara, pois aumenta o custo de encapsulamento do sistema. Uma alternativa é embutir no próprio sistema circuitos de auto-teste, chamado de BIST (Built-In Self-Test) (BARDELL, 1987; STROUD, 2002), que utiliza hardware embarcado para gerar padrões e avaliar respostas. Esta abordagem reduz as restrições de um testador, reduzindo seu custo, e também pode aumentar a utilização da NoC, reduzindo o tempo de teste. Porém, o BIST possui algumas desvantagens: (i) baixa cobertura de falhas em circuitos resistentes a padrões pseudo-aleatórios; (ii) alto consumo de potência devido à alta atividade e chaveamento do circuito; (iii) acréscimo de área de hardware e degradação de desempenho.

Uma alternativa a técnicas de teste baseadas em hardware como BIST é o reuso de processadores embarcados para teste, onde um processador executa um software que testa partes do sistema (KRSTIC et al., 2002; LAI; CHENG, 2001). Esta abordagem possui algumas vantagens como reduzido ou nulo acréscimo de área de hardware e degradação de desempenho. Porém, uma desvantagem é o maior tempo de teste devido à execução serializada do software de teste. Esta desvantagem pode ser minimizada se o sistema possuir diversos processadores embarcados que executem em paralelo um software de teste. Uma NoC composta por múltiplos processadores atribuiria ao sistema características como reuso e escalabilidade, além de uma grande flexibilidade devido à programabilidade da mesma. Entretanto, antes de usar a NoC para transportar dados de teste, é importante que a própria NoC esteja livre de falhas, portanto, a NoC precisa ser testada antes de ser usada. O custo teste da NoC pode ser reduzido se forem consideradas algumas de suas características, como por exemplo a sua regularidade.

## Motivações e Objetivos

A pesquisa em projeto de NoCs é algo recente, mas tem crescido de importância, incluindo centro de pesquisas não somente acadêmicos (DALLY; TOWLES, 2001; BENINI; DE MICHELI, 2002) como também industriais (GOOSSENS; DIELISSEN; RADULESCU, 2005; ARTERIS, 2005). Entretanto, ainda não existem produtos usando esta tecnologia, mas vários pesquisadores estão anunciando os primeiros test-chip. A medida que produtos comecem a ser fabricados em escala comercial, o problema de testar sistemas com varias dezenas de núcleos em uma NoC será evidenciado. Inicialmente as técnicas tradicionais de teste de SoCs poderão ser utilizadas, mas elas acarretam em custos desnecessários que podem ser eliminados com um estudo adequado das características e arquitetura dos sistemas baseados em NoCs. O objetivo desta tese é definir métodos que reduzam os custos de teste de circuitos baseados em NoCs. Para isto, este trabalho foi dividido em três etapas: (i) o reuso da NoC para transporte de dados de teste; (ii) o reuso de processadores embutidos para teste; (iii) o teste da NoC.

## Desenvolvimento, Resultados e Discussão

Alguns atuais sistemas integrados usam de 10 a 15 processadores e, de acordo com (HENKEL, 2003), há uma tendência ao uso de até centenas de processadores heterogêneos conectados em uma NoC. Considerando esta tendência e a dificuldade de testar um sistema desta complexidade, o reuso para teste de ambos NoC e processadores embarcados surge como uma abordagem interessante para reduzir os custos de teste. Os múltiplos caminhos que uma NoC disponibiliza para acessar um núcleo pode ser usado para transporte de dados de teste, e os processadores existentes podem ser programados para rodar programas que testam partes do sistema, aumentando o paralelismo do teste e reduzindo o tempo total de teste.

A estratégia de teste usada nesta tese é baseada no trabalho de Erika e colaboradores (COTA et al., 2002; COTA; LIU, 2006). Neste trabalho, vários detalhes de implementação da NoC são modelados com precisão temporal. Alguns dos tipos de detalhes incluem detalhes de organização da NoC como topologia, aridade de roteadores; detalhes de arquitetura como estratégia de buferização na entrada ou na saída, profundidade de buffers; detalhes funcionais como algoritmos de roteamento, controle de fluxo, arbitração; detalhes temporais como tempo necessário para executar um roteamento, tempo necessário para fazer transferência de um roteador ao outro.

Para que uma nova implementação de NoC fosse suportada por esse modelo, muitas vezes era necessário estender o código fonte para, por exemplo, suportar uma nova topologia de roteadores. Isto torna a técnica pouco abrangente. Entretanto, existem trabalhos que estão inserindo mecanismos que provêem serviços com garantias de qualidade na comunicação realizada pela NoC (GOOSSENS et al., 2002; BJERREGAARD; SPARSO, 2005b). A inserção de garantias de qualidade é importante pois em uma NoC podem existir vários recursos compartilhados entre diferentes canais de comunicação, trazendo indeterminismo no tempo de entrega de dados, que não é aceitável para aplicações com restrições de tempo real. Esta tese prope o uso deste tipo de serviço com garantias para facilitar o teste, uma vez que, por exemplo, pode-se estabelecer uma taxa mínima para envio de padrões de teste. Uma vez que esta garantia de entrega de dados existe em uma comunicação fim-a-fim, detalhes sobre a organização interna da rede podem ser ignorados, oque simplifica o modelo e facilita a aplicação da técnica para diferentes implementações de NoCs com suporte a garantias de serviço.

Outros detalhes a respeito do reuso da NoC para teste, como *o projeto do wrapper dos núcleos do sistema*, também foram estudados. A função dos wrappers é isolar o núcleo enquanto está sendo testado e prover acesso a todos terminais de entrada e saída do mesmo. Mas um problema não abordado nos trabalhos anteriores é que durante o teste parte do sistema está em modo funcional (a NoC) e outra parte em modo de teste (o CUT). Os terminais do CUT são a barreira entre o modo funcional e o modo teste. Entretanto, estes terminais implementam um protocolo de comunicação entre o núcleo e a NoC. Uma vez que o núcleo está em modo de teste, este não pode reagir ao protocolo, portanto, não pode enviar nem receber dados da NoC. Sendo assim, *é necessário que o wrapper tenha a função adicional de implementar ao menos uma versão simplificada do protocolo de comunicação de forma que o fluxo de dados de teste seja mantido.*

Estudando a implementação mínima para três dos mais populares protocolos, AXI, OCP e DTL, concluímos que a funcionalidade dos terminais de entrada pode ser ignorada, e a funcionalidade dos terminais de saída necessita implementar alguma lógica. É função do integrador do sistema conhecer o protocolo e descobrir os valores adequados para cada terminal de saída, uma vez que estes valores podem mudar de acordo com que o núcleo é integrado no sistema. O projetista do núcleo deve prover uma documentação ilustrando todos os modos de uso da porta. Uma vez que os valores corretos são atribuídos aos terminais, o projeto do wrapper é basicamente fazer uma atribuição adequada de célula de wrapper aos terminais de saída. Terminais que não implementam lógica relacionada ao protocolo de comunicação devem receber a célula de wrapper padrão. O problema de conectar cada célula do wrapper as cadeias de varredura internas ao núcleo de forma a reduzir o tempo de teste é idêntico ao problema definido por Marinissen e colaboradores (MARINISSEN et al., 1998) com a diferença que se necessita de conversores paralelo-serial-paralelo que garantam a disponibilidade de um novo dado de teste a cada ciclo de relógio do núcleo.

Uma vez que o projeto do wrapper de cada núcleo esteja definido, a próxima etapa deve realizar o *escalonamento de teste do sistema* de forma a minimizar o tempo de teste total. É necessário estabelecer a quantidade de largura de banda que cada núcleo vai receber para o seu teste. Adaptamos o algoritmo proposto em (GOEL; MARINISSEN, 2003a) para funcionar em um sistema baseado em NoC ao invés de TAM dedicada. Os resultados mostraram que o uso de NoC implica em um pequeno aumento no tempo de teste, mas coma grande vantagem de não requerer TAMs dedicadas.

Os trabalhos que reusam a NoC para transportar dados de teste pressupõem que a mesma foi previamente testada. Entretanto, não existem formas eficientes para *testar uma NoC*. Algumas de suas características fazem com que métodos tradicionais de teste de núcleos de hardware, quando aplicados a uma NoC, torne o custo em área do wrapper muito elevado. Por exemplo, geralmente assume-se que todos núcleos possuem full-scan e células de scan para todos terminais do núcleo. Uma NoC possui dezenas ou centenas de buffers geralmente implementados como um registrador de deslocamento. A implementação automática de cadeias de varredura nestes buffers é um gasto em área desnecessário, pois os buffers já implementam registradores de deslocamento. Outra característica importante é que uma NoC possui um número muito superior de terminais comparado com outros núcleos, uma vez que a NoC pode ser vista como um núcleo que tem conexão com todos núcleos do sistema. Isto requer um número grande de células de wrapper, aumentado a área do sistema. Entretanto, analisando a arquitetura tradicional de uma NoC, é possível utilizar algumas palavras dos buffers dos roteadores para controlar entrada e observar saídas. A última característica importante é que a maioria das implementações de NoCs se baseiam em topologias regulares como mesh e torus. Desta forma, os roteadores podem ser idênticos. A existência de módulos idênticos no sistema pode permitir que um padrão de teste seja enviado em paralelo para vários módulos, ao invés de um só. Isso aumenta o paralelismo do teste, diminuindo o tempo de teste do sistema. Além disto diminui o volume de dados de teste uma vez que o mesmo padrão é enviado para vários módulos e todos os módulos possuem a mesma resposta esperada. Baseando-se nestas três observações, foi desenvolvido um método de teste específico para NoCs, que reduz a área necessária para DfT ao mesmo tempo que reduz o seu tempo de teste e volume de dados de teste. Tomamos por exemplo uma NoC hipotética onde cada porta da NoC é conectada a um núcleo. Uma das vantagens de uma NoC em relação a barramento é justamente poder suportar mais núcleos. Considerando um canal de dados de 32 bits, cada uma destas portas pode ter cerca de 40 a cerca de uma centena de terminais, dependendo do protocolo utilizado e de como este protocolo está configurado. Pode-se chegar a uma quantidade de milhares de terminais a uma NoC. Usando o método tradicional de projeto de wrappers, isto implica que o wrapper teria uma célula para cada um destes terminais, aumentando o custo em área de DfT. Entretanto, observando-se a lógica das portas de entrada de uma roteador existem buffers para armazenamento temporário de dado antes de ser realizado roteamento do mesmo. Os primeiros elementos de memória de cada um destes buffers podem ser usados como um substituto de células de wrapper com a simples adição de cadeias de varredura somente na primeira palavra de cada buffer, como ilustrado na Figura 6. Com esta técnica, ambos o controle das portas de entrada e a observabilidade das portas de saída estão garantidos. A observabilidade também está garantida porque, como as portas de entradas de um roteador estão conectados na saída de outros roteadores, é possível observar a saída de um roteador nas entradas de seus roteadores vizinhos.

Existe uma forte tendência ao uso de redes com topologias regulares devido a simplicidade de seu projeto. Esta regularidade da rede pode trazer benefícios para o teste da mesma de forma que os padrões de teste podem ser enviados em paralelo para todos os roteadores idêntico da rede. Para possibilitar este envio em paralelo de padrões de teste, é necessário um projeto diferenciado das cadeias internas de cada roteador e também de um novo projeto de wrapper. No nível da NoC é necessário que todas as portas de entrada de dados de scan estejam conectadas ao mesmo terminal. As saídas destas cadeias podem ser conectadas a comparadores. Uma vez que todos os roteadores recebem os mesmos estímulos, se todos não tiverem falhas, eles devem gerar as mesmas saídas. Caso contrario, algum dos roteadores tem alguma falha. O comparador proposto possui inclusive suporte a diagnostico para que, no caso de falha, possa ser identificado qual roteador falhou.

Também é necessário garantir que todos os roteadores recebam exatamente os mesmo estímulos nas entradas primárias. O novo wrapper foi projetado de forma a enviar os mesmos estímulos, carregados nas células ci, para cada entrada primárias de cada roteador. O wrapper também possui comparadores nas saídas primárias, cujo resultado da comparação é carregado nas células co. Este projeto de wrapper que requer um número menor de células de wrapper não somente reduz a área, pois não requer uma célula para cada entrada e saída primária da NoC, mas também reduz o tempo de teste uma vez que o comprimento da cadeias externas são reduzidos em comparação com o projeto tradicional de wrapper.

Esta tese também propôs um método de planejamento de teste capaz de *reutilizar os processadores disponíveis no sistema como mecanismos de geração e avaliação de* *teste*, e uma NoC como mecanismo de acesso de teste aos núcleos de hardware de um SoC. O tempo de teste do sistema é avaliado considerando diferentes números de processadores e interfaces externas usadas para teste. Os resultados apresentados utilizando um conjunto de exemplos de sistemas industriais demonstraram que o uso conjunto de NoC e processadores embutidos podem reduzir o tempo de teste sem aumentar a área e reduzir o desempenho do sistema. A descrição original dos estudos de caso utilizados não possui processadores. Desta forma, avaliamos tais estudos de caso com a inclusão de dois diferentes processadores comerciais, chamados de MIPS e SPARC. Programas de teste foram desenvolvidos para estes processadores. Foram avaliados o consumo de potência e o tempo para gerar cada padrão de teste. Estes dois dados foram incluídos em uma ferramenta de escalonamento de teste que explora o paralelismo da NoC para possibilitar o teste de vários núcleos ao mesmo tempo.

## Conclusões

As principais conclusões desta tese são:

Teste baseado em software minimiza os dois principais problemas relacionados ao uso de BIST: perda de desempenho e aumento da área de hardware. A minimização destes problemas possibilita o auto-teste de circuitos de alto desempenho e de difíceis restrições de concepção, tipicamente restrições de freqência de operação e área de hardware. Também se observou que teste baseado em software reduz o tempo de teste de sistemas com NoCs, uma vez que diferente de barramentos, NoCs suportam teste de vários núcleos em paralelo. Porém, novos desafios incluem a geração de vetores de teste com boa cobertura de falhas e área de memória necessária para o programa de teste. Os trabalhos realizados apresentaram programas de geração de padrões de teste aleatórios quem executam rapidamente em processadores como SPARC e MIPS. Entretanto, números aleatórios geralmente apresentam uma baixa cobertura de falhas. é necessário pesquisar novos programas de teste que sejam rápidos de executar e que produzam bons vetores de teste. Uma revisão bibliográfica apontou a existência de códigos de compressão de dados como um possível compromisso entre tempo de execução e qualidade e compressão dos vetores. Na revisão bibliográfica apresentada no relatório do ano anterior não foi encontrada nenhuma abordagem que avaliasse teste baseado em software em NoCs, demonstrando a originalidade do trabalho. No ano de 2004, publicamos o primeiro artigo sobre a temática, apresentando alguns resultados preliminares. Neste ano de 2005 apresentamos trabalhos adicionais sobre reuso da NoC para teste, onde processadores de relevância industrias foram utilizados como estudos de caso, ao invés de processadores educacionais como no artigo anterior. Conclui-se que processadores como MIPS e SPARC podem executar programas de teste mais eficientemente que nos processadores anteriores, demonstrando que a técnica pode ser aplicada em sistemas reais.

Foi desenvolvida uma técnica de teste para NoCs regulares que explora esta regularidade para reduzir os custos de teste. Padrões de teste são enviados em paralelo para todos roteadores idênticos e as saídas são comparadas por comparadores embutidos no sistema. Mostrou-se que tempo de teste, volume de vetores de teste podem ser reduzidos utilizando este técnica. Entretanto é necessário avaliar o impacto em termos de área de roteamento na adição de comparadores no sistema. Como trabalho futuro é necessário pesquisar estratégia de teste para redes irregulares.

Também durante esta tese, repensamos a estratégia de reuso da NoC como TAM. A estratégia anterior exige uma série de detalhes de implementação da NoC de forma a possibilitar uma predição exata do escalonamento dos pacotes de teste. Isto impossibilita o uso da técnica em implementações diferentes de redes, reduzindo a abrangeria do trabalho. Os últimos trabalhos realizados trabalhou-se com NoCs que oferecem qualidade de serviço. Estas NoCs garantem a entrega de pacotes em intervalos de tempo predeterminado, não necessitando modelar detalhes de implementação da NoC, possibilitando a aplicação da técnica de reuso da NoC para teste em diversas implementações de NoCs. Finalmente, comparamos a arquitetura de teste proposta com a arquitetura de teste convencional baseada em TAMs dedicadas estabelecendo assim as reais vantagens e desvantagens de ambos os métodos.

Por fim, a maior contribuição desta tese foi pesquisar o problema de teste em sistemas baseados em NoC com um enfoque que favorecesse a generalidade do método, de forma que esse possa ser aplicado facilmente a um maior número de NoCs.

# APPENDIX B

## List of Papers

During the period comprising this thesis, the following articles have been published:

1. **IEEE TC**: Cota, E.F.; Kastensmidt, F.L.; Cassel, M.; Hervé, M.; Almeida, P.; Meirelles, P.; Amory, A.M.; Lubaszewski, M.S. “A High Fault Coverage Approach for the Test of Data, Control and Handshake Interconnects in Mesh Networks-on-Chip”. Journal IEEE Transaction on Computers, to be published on 2008.
2. **IEE CDT**: Amory, A.M.; Goossens, K.; Marinissen, E.J.; Lubaszewski, M.; Moraes, F.G. “Wrapper Design for the Reuse of NOCs and Other Functional Interconnects as Test Access Mechanism”. Journal IEE Proceedings Computers and Digital Techniques. v.1, pp. 197–206, 2007.
3. **VTS07**: Amory, A.M.; Ferlini, F.; Lubaszewski, M.; Moraes, F.G. “Design-for-Test for Routers Based on Reuse of Networks-on-Chip as Test Access Mechanism”. pp. 435–440. 2007.
4. **ITC07**: Kastensmidt, F.L.; Cota, E.F.; Cassel, M.; Meirelles, P.; Lubaszewski, M.S.; Amory, A.M “Redefining and Testing Interconnect Faults in Mesh NoCs”. pp. 1–10. 2007.
5. **ETS06**: Amory, A.M.; Goossens, K.; Marinissen, E.J.; Lubaszewski, M.; Moraes, F.G. “Wrapper Design for the Reuse of Networks-on-Chip as Test Access Mechanism”. pp. 213-218. 2006.
6. **ITC05**: Amory, A.M.; Brião, E.W.; Cota, E.F.; Lubaszewski, M.; Moraes, F.G. “A Scalable Test Strategy for Network-on-Chip Routers”. pp. 591–599. 2005.
7. **ETS05**: Amory, A.M.; Brião, E.W.; Cota, E.F.; Lubaszewski, M.; Moraes, F.G. “A Cost-Effective Test Flow for Homogeneous Network-on-Chip”. poster. 2005.
8. **DATE05**: Amory, A.M.; Lubaszewski, M.; Moraes, F.G.; Moreno, E.I. “Test Time Reduction Reusing Multiple Processors in a Network-on-Chip Based Architecture”. pp. 62–63 (short paper). 2005.
9. **SBCCI04**: Amory, A.M.; Cota, E.F.; Lubaszewski, M.; Moraes, F.G. “Reducing Test Time with Processor Reuse in Network-on-Chip Based Systems”. pp. 111–116. 2004.
10. **LATW04**: Amory, A.M.; Lubaszewski, M.; Moraes, F.G. “A Programmable Logic BIST Controller for IP Cores”. pp. 104–109. 2004.

11. **LATW04**: Marcon, C.A.M.; Amory, A.M.; Lubaszewski, M.; Susin, A.A.; Calazans, N.L.V.; Moraes, F.G. “Applying Memory Test Algorithms to Embedded Systems”. pp. 43–48. 2004.