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

ÉRIKA FERNANDES COTA

Reuse-based Test Planning for Core-based Systems-on-chip

Thesis presented in partial fulfillment
of the requirements for the degree of
Doctor of Computer Science

Prof. Dr. Marcelo Lubaszewski
Advisor

Porto Alegre, September 2003

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Cota, Érika Fernandes

Reuse-based Test Planning for Core-based Systems-on-chip / Érika Fernandes Cota. – Porto Alegre: Programa de Pós-Graduação em Computação, 2003.

166 f.: il.

Thesis (Ph.D.) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2003. Advisor: Marcelo Lubaszewski.

1. SoC testing. 2. Testing of embedded cores. 3. Design for test. 4. Design space exploration. 5. Network-on-chip. I. Lubaszewski, Marcelo. II. Título.

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
Reitora: Prof[a]. Wrana Maria Panizzi
Pró-Reitor de Ensino: Prof. José Carlos Ferraz Hennemann
Pró-Reitora Adjunta de Pós-Graduação: Profa. Jocélia Grazia
Diretor do Instituto de Informática: Prof. Philippe Olivier Alexandre Navaux
Coordenador do PPGC: Prof. Carlos Alberto Heuser
Bibliotecária-chefe do Instituto de Informática: Beatriz Regina Bastos Haro

“yada yada,
yada....”

ACKNOWLEDGMENTS

I am beholden to my advisor, Dr. Marcelo Lubaszewski, for his encouraging discussions, unconditional support, and trust during all these years. I also want to express my gratitude to Dr. Luigi Carro for his questioning and passionate attitude and inspiring suggestions, remembering me what research is all about. It has been a pleasure to collaborate with Dr. Altamiro Susin, who reminds me how important it is to let the apparently improbable ideas grow.

From the test research community, I am indebted to the numerous researches whom I had the opportunity to work with: Dr. Alex Orailoglu from the Reliable Systems Synthesis Laboratory of UCSD, for the technical discussions and for showing me other aspects of the research; Erik Jan Marinissen, from Philips, for his friendship and encouragement. It has been a pleasure to collaborate with Dr. Michel Renovell, Dr. Florence Azaïs, Dr. Bruno Rouzeyre, Dr. Yves Bertrand, among others from the LIRMM Laboratory. I have enjoyed to collaborate in different projects with Dr. Raoul Velazco and Dr. Salvador Mir, from TIMA Laboratory.

From home, I am immensely grateful to my "Little Trem", Luigi, for his infinite patience (considering his explosive mood...) and unrestricted support; I cannot say enough thanks to my parents, José Cota and Maristella, for all their history and effort that brought me up to this point; thanks to my parents-in-law, Cesare and Esther, for their affection and help during this time.

It has been a pleasure to share the office with my colleagues of Ph.D. and M.Sc.: Márcio Kreutz, César Zeferino, Margrit Krug, Júlio Mattos, Lisane de Brisolara, Marcelo Negreiros, Adão Júnior, José Güntzel, and Marcelo Johann. I am grateful for their support and friendship. I am also grateful to Fernanda Lima and Zingara, who became dear friends in the last (tough) years.

Thanks are due to the Informatics Institute staff: Luis Otávio and his team: Margareth, Jorge, Júnior, Leandro, and Elgio, for the infra-structure; Bia and the library team: Ida, Henrique, and Adriana; Eliane, Sr. Astro, and the security people; Silvana and the administrative team: Lourdes, Jorge, Schneider, Elisiane, Angela, Claudia, and Alex, also for the infra-structure;

I also want to thank some students that have worked with me in the last few years, and that helped generating some of the results presented in this thesis: Lisane de Brisolara, Cristiano Lazzari, Leandro Cassol, Eduardo Back, Rodrigo Boccasius, Renato Hentschk, and Guilherme Schneider.

I want to express my gratitude to the members of the jury of my thesis proposal, Erik Jan Marinissen, Dr. Raoul Velazco, Dr. Altamiro Susin, and Dr. Ingrid Jansch Porto, for their comments during the definition of this thesis. Finally, I want to thank the members of the final jury, Dr. Antônio Otávio Fernandes, Dr. Marius Strum, and Dr. Altamiro

Susin, for their participation and evaluation.

I acknowledge the work of CNPq and CAPES for the scholarships during this Ph.D and the "sandwich" internship at San Diego, respectively. Finally, I am once again grateful to the brazilian taxpayers (more than never, the real ones) that financed this work.

CONTENTS

LIST OF ABBREVIATIONS AND ACRONYMS 9
LIST OF FIGURES 11
LIST OF TABLES 13
ABSTRACT 15
RESUMO 17
1 INTRODUCTION 19
2 SOC DESIGN TRENDS AND TEST CHALLENGES 23
2.1 Reuse-based SoC Design 23
2.2 NoC-based SoC Design 25
2.3 Test Challenges in SoC Design 28
2.3.1 Core Test Requirements 28
2.3.2 Interconnection Requirements 30
2.3.3 System Requirements 30
3 RELATED WORKS AND MOTIVATIONS 33
3.1 Test Access Mechanism Definition 33
3.2 Test Scheduling Definition 40
3.3 Test Planning 42
3.4 Test Standard Initiatives 43
3.4.1 IEEE P1500 Standard 43
3.4.2 VSI Alliance 45
3.4.3 ITC'02 SoC Test Benchmarks 45
3.5 Contributions of this Work 46
4 TEST PLANNING AND DESIGN SPACE EXPLORATION IN CORE-BASED SYSTEMS 49
4.1 TAM Definition and System Cost Factors 50
4.1.1 Direct External Access 51
4.1.2 Reuse of Functional Connections 52
4.1.3 Use of Serial Bypass 53
4.1.4 Use of Transparency Functions 53
4.1.5 Parallel Bypass 54
4.2 Problem Statement 55

4.3 Solution Modeling . . . 56
4.3.1 Initial Approach . . . 56
4.3.2 Final Approach . . . 57
4.4 The Proposed Heuristic . . . 61
4.5 Heuristic Complexity . . . 63
4.6 Experimental Setup . . . 64
4.6.1 ITC'02 SoC Test Benchmarks . . . 64
4.7 Experimental Results . . . 66
4.7.1 Benchmarks d695, g1023, f2126, q12710, and t512505 . . . 66
4.7.2 Benchmark h953 . . . 73
4.7.3 Benchmarks u226 and d281 . . . 75
4.7.4 Benchmarks p22810, p34392, p93791, and a586710 . . . 80
4.8 System Characteristics and Benchmark format . . . 88

5 NOC-BASED TESTING OF CORE-BASED SYSTEMS-ON-CHIP . . . 93
5.1 Using the NoC During Test . . . 96
5.1.1 Exploiting pipeline within the NoC . . . 99
5.1.2 Power Consumption Calculation . . . 100
5.1.3 Power-Aware Test Scheduling . . . 101
5.1.4 Example . . . 104
5.2 Complexity Analysis . . . 106
5.3 Experimental Results . . . 107
5.3.1 Benchmarks d695, g1023, f2126, q12710, and t512505 . . . 108
5.3.2 Benchmark h953 . . . 116
5.3.3 Benchmarks u226 and d281 . . . 116
5.3.4 Benchmarks p22810, p34392, p93791, and a586710 . . . 121
5.4 System Configurations and Resulting Test Time . . . 127
5.4.1 Placement of the Cores in the Network . . . 127
5.4.2 Number of Interfaces with the Tester . . . 127
5.4.3 Network Power Profile . . . 129

6 DISCUSSION . . . 135
6.1 Reuse-based versus NoC-based Test Planning . . . 135
6.2 Reuse-based versus Bus-based Test Planning . . . 137
6.3 Limitations of the Proposed Methods . . . 141

7 FINAL REMARKS . . . 143

REFERENCES . . . 147

APPENDIX A PLANEJAMENTO DE TESTE BASEADO EM REUSO PARA SISTEMAS EM SIL{I . . . 159

APPENDIX B CD-ROM DESCRIPTION . . . 165

LIST OF ABBREVIATIONS AND ACRONYMS

API	Application Programming Interface
ATE	Automatic Test Equipment
BIST	Built-In Self Test
CAD	Computer-Aided Design
CTL	Core Test Language
CUT	Circuit under test
DFT	Design for test
HDL	Hardware Description Language
IP	Intellectual Property
ITRS	International Technology Roadmap for Semiconductors
NoC	Network-on-chip
P1500 SECT	P1500 Standard for Embedded Core Test
RTL	Register Transfer Level
RTOS	Real Time Operating System
SIA	Semiconductor Industry Association
SoB	System-on-board
SoC	System-on-chip
TAM	Test Access Mechanism
TAP	Test Access Port
STIL	Standard Test Interface Language
UDL	User defined logic
VSIA	Virtual Socket Interface Alliance

LIST OF FIGURES

Figure 2.1: Core types (KUCUKCAKAR, 1998) 24
Figure 2.2: Example of SoC with super-cores and UDL (ZORIAN, 1997) 25
Figure 2.3: Some NoC topologies 27
Figure 2.4: Differences between SoB and SoC testing (MARINISSEN; ZORIAN, 1999) 29

Figure 3.1: Conceptual architecture for the SoC testing (ZORIAN; MARINISSEN; DEY, 1998) 34
Figure 3.2: Relationship between TAM width and test costs 35
Figure 3.3: Test Bus Architectures (AERTS; MARINISSEN, 1998) 37
Figure 3.4: Test Rail Achitecture (MARINISSEN et al., 1998) 38
Figure 3.5: Flexible-width test bus architecture (IYENGAR; CHAKRABARTY; MARINISSEN, 2002a) 38
Figure 3.6: P1500 wrapper structure (MARINISSEN et al., 2002) 44

Figure 4.1: Core interface 50
Figure 4.2: Global TAM for a CUT 51
Figure 4.3: Direct external access TAM 52
Figure 4.4: Access via functional connections 52
Figure 4.5: Access via serial bypass 53
Figure 4.6: Access via transparent core 54
Figure 4.7: Access via Parallel Bypass 54
Figure 4.8: Test schedule for minimum test time 58
Figure 4.9: Schedule Construction 59
Figure 4.10: Tree Construction 60
Figure 4.11: Pseudo-code of proposed model 62
Figure 4.12: Floorplanning of benchmark d695 67
Figure 4.13: Floorplanning of benchmark g1023 69
Figure 4.14: Floorplanning of benchmark f2126 71
Figure 4.15: Floorplanning of benchmark q12710 72
Figure 4.16: Floorplanning of benchmark t512505 73
Figure 4.17: Floorplanning of benchmark h953 75
Figure 4.18: Floorplanning of benchmark u226 77
Figure 4.19: Floorplanning of benchmark d281 79
Figure 4.20: Floorplanning of benchmark p22810 82
Figure 4.21: Floorplanning of benchmark p34392 83
Figure 4.22: Floorplanning of benchmark p93791 85
Figure 4.23: Floorplanning of benchmark a586710 85

Figure 4.24: Two configurations for system d695 88

Figure 5.1: Basic structure of the SOCIN router (ZEFERINO; SUSIN, 2003) . . 94
Figure 5.2: SOCIN topologies 94
Figure 5.3: System d695 implemented in a 4x3 grid SOCIN NoC 95
Figure 5.4: Wrapper configurations 97
Figure 5.5: Pseudo-code of the adapted list-scheduling algorithm 103
Figure 5.6: Scheduling process considering power constraints 105
Figure 5.7: System g1023 implemented in a 4x4 NoC 109
Figure 5.8: System f2126 implemented in a 2x2 NoC 111
Figure 5.9: System q12710 implemented in a 2x2 NoC 111
Figure 5.10: System t512505 implemented in a 5x7 NoC 115
Figure 5.11: System h953 implemented in a 2x4 NoC 116
Figure 5.12: System u226 implemented in a 3x3 NoC 117
Figure 5.13: System d281 implemented in a 2x4 NoC 118
Figure 5.14: System p22810 implemented in a 4x6 NoC 121
Figure 5.15: System p34392 implemented in a 2x3 NoC 121
Figure 5.16: System p93791 implemented in a 3x5 NoC 122
Figure 5.17: System a586710 implemented in a 2x2 NoC 124
Figure 5.18: Different placements for system d695 in a 3x4 network 128
Figure 5.19: d695 test time variation with the number of interfaces with the tester . 128

LIST OF TABLES

Table 4.1: Some Characteristics of the ITC'02 Benchmarks (MARINISSEN; IYENGAR; CHAKRABARTY, 2002) . . . 65
Table 4.2: Test requirements of benchmark d695 . . . 68
Table 4.3: Test results for benchmark d695 . . . 68
Table 4.4: Test requirements of benchmark g1023 . . . 70
Table 4.5: Test results for benchmark g1023 . . . 70
Table 4.6: Test requirements of benchmark f2126 . . . 71
Table 4.7: Test results for benchmark f2126 . . . 71
Table 4.8: Test requirements of benchmark q12710 . . . 72
Table 4.9: Test results for benchmark q12710 . . . 73
Table 4.10: Test requirements of benchmark t512505 . . . 74
Table 4.11: Test results for benchmark t512505 . . . 75
Table 4.12: Test requirements of benchmark h953 . . . 76
Table 4.13: Test results for benchmark h953 . . . 76
Table 4.14: Test requirements of benchmark u226 . . . 78
Table 4.15: Test results for benchmark u226 . . . 78
Table 4.16: Test requirements of benchmark d281 . . . 79
Table 4.17: Test results for benchmark d281 . . . 80
Table 4.18: Test requirements of benchmark p22810 . . . 81
Table 4.19: Test results for benchmark p22810 . . . 82
Table 4.20: Test requirements of benchmark p34392 . . . 83
Table 4.21: Test results for benchmark p34392 . . . 84
Table 4.22: Test requirements of benchmark p93791 . . . 86
Table 4.23: Test results for benchmark p93791 . . . 87
Table 4.24: Test requirements of benchmark a586710 . . . 87
Table 4.25: Test results for benchmark a586710 . . . 87
Table 4.26: New test planning results for d695 . . . 89
Table 4.27: New test planning results for p22810 . . . 89
Table 4.28: New test planning results for p93791 . . . 90
Table 4.29: New test planning results for u226 . . . 90

Table 5.1: Test packets for system d695 . . . 98
Table 5.2: Test results for benchmark d695 . . . 108
Table 5.3: Test packets for system g1023 . . . 109
Table 5.4: Test results for benchmark g1023 . . . 110
Table 5.5: Test packets for system f2126 . . . 111
Table 5.6: Test results for benchmark f2126 . . . 112

Table 5.7: Test packets for system q12710 112
Table 5.8: Test results for benchmark q12710 113
Table 5.9: Test packets for system t512505 114
Table 5.10: Test results for benchmark t512505 115
Table 5.11: Test packets for system h953 116
Table 5.12: Test results for benchmark h953 117
Table 5.13: Test packets for system u226 118
Table 5.14: Test results for benchmark u226 119
Table 5.15: Test packets for system d281 119
Table 5.16: Test results for benchmark d281 120
Table 5.17: Test packets for system p22810 122
Table 5.18: Test results for benchmark p22810 123
Table 5.19: Test packets for system p34392 123
Table 5.20: Test results for benchmark p34392 124
Table 5.21: Test packets for system p93791 125
Table 5.22: Test results for benchmark p93791 126
Table 5.23: Test packets for system a586710 126
Table 5.24: d695 test time for different placements in the network 127
Table 5.25: Test time of t512505 for different interface configurations 129
Table 5.26: Test times for d695: cores consumption $>>$ routers consumption . . 130
Table 5.27: Test times for g1023: cores consumption $>>$ routers consumption . . 130
Table 5.28: Test times for p22810: cores consumption $>>$ routers consumption . 131
Table 5.29: Test times for d695: cores consumption$\approx$routers consumption 132
Table 5.30: Test times for g1023: cores consumption$\approx$routers consumption . . . 132
Table 5.31: Test times for p22810: cores consumption$\approx$routers consumption . . . 132
Table 5.32: Test times for p93791: 5 inputs and 4 outputs 133

Table 6.1: Comparative results between proposed approaches 136
Table 6.2: Comparative results with bus-based methods for W = 32 139

ABSTRACT

Electronic applications are currently developed under the reuse-based paradigm. This design methodology presents several advantages for the reduction of the design complexity, but brings new challenges for the test of the final circuit. The access to embedded cores, the integration of several test methods, and the optimization of the several cost factors are just a few of the several problems that need to be tackled during test planning. Within this context, this thesis proposes two test planning approaches that aim at reducing the test costs of a core-based system by means of hardware reuse and integration of the test planning into the design flow.

The first approach considers systems whose cores are connected directly or through a functional bus. The test planning method consists of a comprehensive model that includes the definition of a multi-mode access mechanism inside the chip and a search algorithm for the exploration of the design space. The access mechanism model considers the reuse of functional connections as well as partial test buses, cores transparency, and other bypass modes. The test schedule is defined in conjunction with the access mechanism so that good trade-offs among the costs of pins, area, and test time can be sought. Furthermore, system power constraints are also considered. This expansion of concerns makes it possible an efficient, yet fine-grained search, in the huge design space of a reuse-based environment. Experimental results clearly show the variety of trade-offs that can be explored using the proposed model, and its effectiveness on optimizing the system test plan.

Networks-on-chip are likely to become the main communication platform of systems-on-chip. Thus, the second approach presented in this work proposes the reuse of the on-chip network for the test of the cores embedded into the systems that use this communication platform. A power-aware test scheduling algorithm aiming at exploiting the network characteristics to minimize the system test time is presented. The reuse strategy is evaluated considering a number of system configurations, such as different positions of the cores in the network, power consumption constraints and number of interfaces with the tester. Experimental results show that the parallelization capability of the network can be exploited to reduce the system test time, whereas area and pin overhead are strongly minimized.

In this manuscript, the main problems of the test of core-based systems are firstly identified and the current solutions are discussed. The problems being tackled by this thesis are then listed and the test planning approaches are detailed. Both test planning techniques are validated for the recently released ITC'02 SoC Test Benchmarks, and further compared to other test planning methods of the literature. This comparison confirms the efficiency of the proposed methods.

Keywords: SoC testing, testing of embedded cores, design for test, design space exploration, network-on-chip.

Planejamento de Teste para Sistemas de Hardware Integrados Baseados em Componentes Virtuais

RESUMO

O projeto de sistemas eletrônicos atuais segue o paradigma do reuso de componentes de hardware. Este paradigma reduz a complexidade do projeto de um *chip*, mas cria novos desafios para o projetista do sistema em relação ao teste do produto final. O acesso aos núcleos profundamente embutidos no sistema, a integração dos diversos métodos de teste e a otimização dos diversos fatores de custo do sistema são alguns dos problemas que precisam ser resolvidos durante o planejamento do teste de produção do novo circuito. Neste contexto, esta tese propõe duas abordagens para o planejamento de teste de sistemas integrados. As abordagens propostas têm como principal objetivo a redução dos custos de teste através do reuso dos recursos de hardware disponíveis no sistema e da integração do planejamento de teste no fluxo de projeto do circuito.

A primeira abordagem considera os sistemas cujos componentes se comunicam através de conexões dedicadas ou barramentos funcionais. O método proposto consiste na definição de um mecanismo de acesso aos componentes do circuito e de um algoritmo para exploração do espaço de projeto. O mecanismo de acesso prevê o reuso das conexões funcionais, o uso de barramentos de teste locais, núcleos transparentes e outros modos de passagem do sinal de teste. O algoritmo de escalonamento de teste é definido juntamente com o mecanismo de acesso, de forma que diferentes combinações de custos sejam exploradas. Além disso, restrições de consumo de potência do sistema podem ser consideradas durante o escalonamento dos testes. Os resultados experimentais apresentados para este método mostram claramente a variedade de soluções que podem ser exploradas e a eficiência desta abordagem na otimização do teste de um sistema complexo.

A segunda abordagem de planejamento de teste propõe o reuso de redes em-chip como mecanismo de acesso aos componentes dos sistemas construídos sobre esta plataforma de comunicação. Um algoritmo de escalonamento de teste que considera as restrições de potência da aplicação é apresentado e a estratégia de teste é avaliada para diferentes configurações do sistema. Os resultados experimentais mostram que a capacidade de paralelização da rede em-chip pode ser explorada para reduzir o tempo de teste do sistema, enquanto os custos de área e pinos de teste são drasticamente minimizados.

Neste manuscrito, os principais problemas relacionados ao teste dos sistemas integrados baseados em componentes virtuais são identificados e as soluções já apresentadas na literatura são discutidas. Em seguida, os problemas tratados por este traballho são listados e as abordagens propostas são detalhadas. Ambas as técnicas são validadas através dos sistemas disponíveis no *ITC'02 SoC Test Benchmarks*. As técnicas propostas são ainda comparadas com outras abordagens de teste apresentadas recentemente. Esta comparação confirma a eficácia dos métodos desenvolvidos nesta tese.

Palavras-chave: teste de sistemas integrados, teste de núcleos de hardware embarcados, projeto visando o teste, exploração do espaço de projeto, redes de interconexão em-chip.

1 INTRODUCTION

With the increasing complexity of current integrated circuits, testing has become one of the most expensive and time-consuming tasks of the circuit design. The density of current systems-on-chip (SoCs) and the paradigm of core-based design have posed important difficulties for the test of the resulting chip, which are mainly related to the information requirements for the definition of the system test plan. For the new systems, important information must flow from the core providers to the system integrators and finally to the test engineers, who are not necessarily part of the same department or company. Today, the cost of testing a SoC is estimated to be as high as 50% of the total cost of the chip (ZORIAN; DEY; RODGERS, 2000) and the reduction of this cost is crucial for the electronic market.

One of the most important problems for the test of a SoC is the access to the embedded cores during test. The increase in the number of metal layers and number of transistors in the same silicon area lead to an increase in the complexity of the electronic systems, which present now a large number of logic blocks deeply embedded into the chip. Such blocks require some type of electronic access during test, since they can not be directly accessed from the system interface. Moreover, such access mechanism is intimately related to the resulting testing costs of the system. For example, a single access mechanism that is shared among all embedded cores reduces the costs in terms of area overhead for the system, but leads to a large test time, since all cores must be tested serially. On the other hand, an exclusive access mechanism for each core results in a very reduced test time, but at a possibly unacceptable cost in terms of area overhead and pin count or, even, power consumption. Therefore, a great deal of effort has been spent in the last few years for the development of cost-effective test techniques for core-based SoCs, all of them aiming at the reduction of the test cost and complexity.

Despite the large number of SoC test approaches proposed in the last few years, one can observe two interesting aspects of such solutions: very few of them reuse the system resources during test, and very few of them consider the integration of the test planning task in the early steps of the SoC design flow. Considering the first aspect, only the first works in SoC testing considered the reuse of system connections and logic for the transmission of test data between an external tester and the embedded cores. The majority of the SoC test solutions nowadays is based on the efficient insertion of test buses. The main reason for this change seems to be the *ad hoc* characteristic of the initial works. In many cases, they assume the modification of the core logic to transmit the test data, which is usually not possible because of the intellectual property (IP) protection of the third part blocks. When the cores are not modified, specific bypass structures around the cores are required. Such structures were not standardized and required an extra effort from the system designer to be implemented. As for the second aspect, because of the complexity

of current systems, the test planning task can no longer be left for the latest stages of the design or the turnaround time of the product can be too high. Usually, the system synthesis already presents a number of problems for the system integrator (BERGAMASCHI et al., 2001): the lack of design tools capable of dealing with the variety of core formats, the lack of a single interface standard to communicate all cores in the system, IP protection issues, the size and complexity of the system, and so on. For most of the test solutions proposed in the literature, the test planning can be defined in parallel with the system design. However, they usually do not consider the system characteristics for the test definition. Only the cores requirements in terms of pins and number of test vectors are considered. Thus, each solution devised by the test plan must be verified by the system integrator separately. On the other hand, the system integrator is not always a test specialist nor is the test engineer a design expert. Moreover, there is a number of other requirements for the system synthesis that must be taken care of. Therefore, test planning tools capable of helping both, test engineers and system integrators to make design decisions at the early stages of the system design are of extreme importance to assure an effective test plan for a SoC. Such tools must take into consideration the system characteristics, such as cores floorplanning, functional connections, an so on, in addition to the cores test requirements. This way, the best trade-off among all costs involved in the system synthesis (chip area, power consumption, pin count, test time, design time, among others) can be found.

In this thesis, two test planning approaches focused on the reuse of the system resources during test and on the design space exploration are proposed. The test planning methods are defined according to the connection model of the SoC and aim at helping the system integrator and the test engineer to evaluate the impact of the test solution on the global system cost, for a number of possible configurations of a SoC.

In the first approach, a SoC with a core-to-core connection model is assumed and the test planning tool is based on two main aspects: 1) it considers a mixed set of access mechanisms that includes the insertion of partial test buses, the reuse of functional connections, the use of available transparency modes of the cores, and other bypass modes available through the wrapper or the core configuration; 2) both the test schedule and the global test access mechanisms are defined together, and not as independent tasks as in other approaches. This aspect allows the exploration of the design space, so that good compromises among the various trade-offs being sought in the system synthesis can be found. The main contribution of this method is the use of several types of access mechanisms and the consideration of different optimization factors (area, pins, test time, and power consumption) during the global test access and test schedule definition. This expansion of concerns is combined with an efficient, yet fine-grained search, in the huge design space of the reuse-based environment.

In the second approach, a system implemented over a communication platform called network-on-chip (NoC) is assumed. For those systems, the communication among the cores is implemented through a switching network that is integrated into the chip. This network already provides a real and efficient access to each block embedded into the circuit. Thus, instead of inserting new buses into the system with the sole purpose of test access, the reuse of the available communication platform during test is proposed and a test scheduling algorithm is developed. This is the first work that systematize the test of cores embedded into a NoC-based system.

The two test planning approaches are explained and formalized, and the prototype tools that implement the proposed methods are presented. Furthermore, the methods are validated using the ITC'02 SoC Test Benchmarks (MARINISSEN; IYENGAR; CHAKRABARTY,

2002), a set of SoC examples provided by both, industry and academia, to ease the comparison among the several SoC test solutions available in the literature. One will observe in the experimental results presented in this dissertation that the proposed reuse-based SoC test techniques are indeed capable of devising a good test plan for the system not only in terms of test cost reduction, but also for the integration of the test planning into the initial steps of the system design.

The sequel of this dissertation is divided as follows: Chapter 2 presents a brief introduction to the terms and current trends of SoC design, such as the core-based paradigm and the new communication model based on networks-on-chip. That chapter also details the main challenges for the test of core-based systems-on-chip. Chapter 3 gives an overview of the several SoC test solutions presented in the last few years, and explains the main motivations for this work. Chapters 4 and 5 explain, respectively, the two reuse-based test planning approaches proposed in this thesis, including the experimental results for the ITC'02 benchmarks. Chapter 6 compares the two approaches to each other, discusses their integration at the system design flow, and compares the proposed methods to other techniques. Chapter 7 concludes this manuscript with some final remarks and future work.

2 SOC DESIGN TRENDS AND TEST CHALLENGES

This work tackles some problems related to the test of complex circuits, usually called core-based systems. Such systems present a design cycle that is quite different from the traditional ASICs. As a consequence, the testing requirements of the core-based systems are also very distinct.

In this chapter, some concepts of the core-based designed are reviewed in Section 2.1 while Section 2.2 discusses the design of core-based systems using interconnection networks. Then, the main problems related to the test of core-based systems are listed in Section 2.3.

2.1 Reuse-based SoC Design

The design and manufacturing of integrated circuits is currently based on the integration of a number of pre-designed intellectual property (IP) blocks, or cores, in a single chip. Although the reuse has always been present in the design of electronic circuits, this practice has been extended and formalized in the last decade, becoming the new design paradigm of the electronic industry. The reuse of previously designed functional blocks is now the key for the design of high performance circuits with large gate counts in a short time (KUCUKCAKAR, 1998). Such a design practice is known as core-based or IP-based design, or simply as System-on-Chip (SoC) design. The main difference between a SoC and a traditional System-on-Board (SoB), which is also based on previously designed parts, is that in the former, all cores are synthesized together in a single chip, whereas in the latter each functional block is synthesized and manufactured separately, and then mounted in a discrete board. Furthermore, the reusable blocks of the SoC are also known as *virtual components*, since they are delivered as a description of a logic rather than a manufactured IC, and this constitutes another important difference between traditional design methods and core-based systems.

In the early days of SoCs, components were not really designed for reuse. However, gradually, component design evolved to include more parameterization and standard interfaces (BERGAMASCHI; COHN, 2002). Current available cores include microprocessors, memories, network interfaces, cryptography circuits, analog interfaces, among others (CMP, 2003). The more IP providers are present in the market, the more functionalities become available, and the more are the advantages of the core reuse. As a consequence, new technologies are incorporated in the products while the design time is reduced.

Embedded systems are a typical application where the core-based design is extensively applied. Cell phones, portable medical equipments, robots, and automotive controllers, are some examples of such systems. The successful design of such complex

single-chip applications requires expertise in a number of technology areas such as signal processing, encryption, and analog and RF designs. These technologies are increasingly hard to find in a single design house (GUPTA; ZORIAN, 1997). Moreover, high performance, reduced power consumption and short time-to-market are common requirements for those applications. Therefore, it is interesting to have all (or most) functional blocks (A/D converters, microprocessors, memories, mixed-signal blocks, and so on) already available. In this business model, the specialists in a specific design model (analog or RF, for example) are the core providers, and the application designer can focus on the system aspects only.

IP components are usually available in three forms (GUPTA; ZORIAN, 1997; KUCUKCAKAR, 1998): hard, firm, and soft cores.

- **Hard cores** are provided as black boxes, usually in layout form and with encrypted simulation model. Due to their high performance and/or design complexity, these cores need to be provided as an optimized layout in a given technology. Examples of hard cores are microprocessors, memories, PLLs, and UARTs (CMP, 2003)

- **Firm cores** are provided as a synthesized netlist, that is, after logic synthesis and technology mapping, but without layout information. Those cores are described in a hardware description language (HDL), which can be simulated and changed if necessary. However, the user does not need to re-synthesize the block. ASICs and FPGAs are some examples of firm cores available in the market.

- **Soft cores** are given as register-transfer level (RTL) HDLs, and the user is responsible for its synthesis and layout. However, the soft cores providers usually supply synthesis and layout scripts, as well as timing assertions to make it easier the integration of those parts into the rest of the system. Some examples of soft cores are DSP blocks, Ethernet controllers, micro-controllers, and DMA controllers (CMP, 2003).

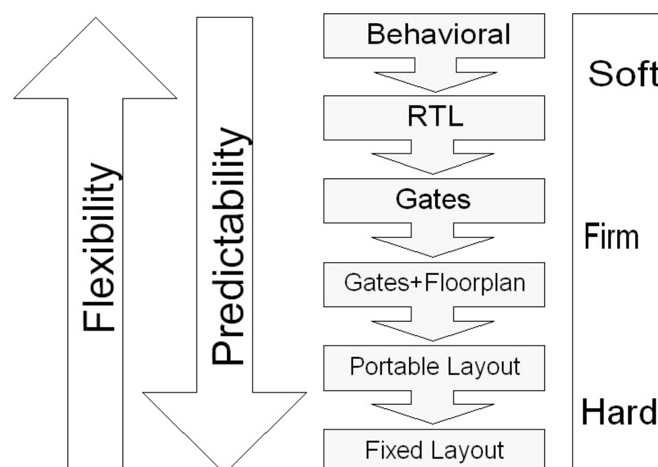


Figure 2.1: Core types (KUCUKCAKAR, 1998)

Figure 2.1 shows the relationship between the core model and its description language. The figure also shows the trade-offs in terms of flexibility and predictability of each type of core (KUCUKCAKAR, 1998). Having a completely rigid and validated layout with definite timing, the hard cores have rapid integration at the expense of flexibility. They

can further create place and route problems due to their rigidity. Technology-mapped gate-level netlists constitute a core style with less predictability than layouts, but allow significant flexibility during place and route. When technology-mapped gates (or logic) and predetermined floorplanning are used, the resulting firm cores are both flexible and more predictable. Properly written RTL sources can be synthesized into most technologies, but the freedom to change timing, area or power is not as great as in the case of behavioral models that can be synthesized via behavioral synthesis (KUCUKCAKAR, 1998).

Figure 2.2 shows an example of a core-based system, where one of the cores is a super-core, that is, it embeds other cores. Notice the presence of a User Defined Logic (UDL) that adds some functionality to the system and is designed by the system integrator.

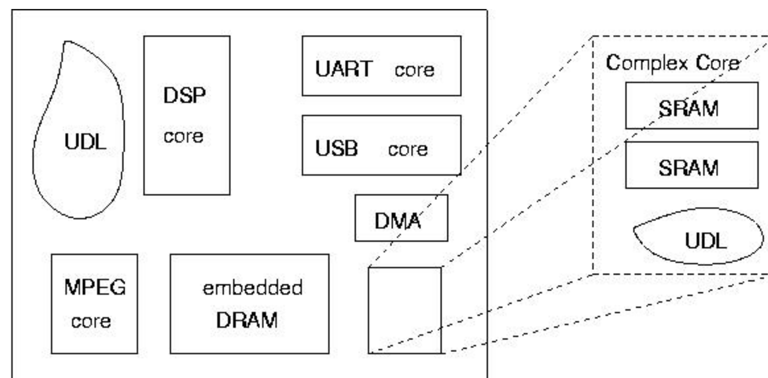


Figure 2.2: Example of SoC with super-cores and UDL (ZORIAN, 1997)

Although the design cycle of a complex system has improved with the advent of the core-based design paradigm, this is still an error-prone, labor-intensive and time-consuming task (BERGAMASCHI et al., 2001). Hence, design methods for SoCs are still an important research subject and a number of design aspects are being considered, such as the expansion of the reuse to a set of blocks and the communication issues. Moreover, the test of such systems became an important problem.

According to Zorian *et.al.* (ZORIAN; DEY; RODGERS, 2000), the effort to generate tests has been growing geometrically along with the product complexity. In the year 2000, the capital costs for testing, based on the 1997 SIA technology roadmap for semiconductors (SIA, 1997), was about 50% of the overall IC cost (ZORIAN; DEY; RODGERS, 2000). This cost showed signs of reduction according to the 1999 ITRS roadmap (SIA, 1999), due to the research and industrial efforts towards the development of cost-effective test solutions for SoCs. However, the costs related to the test of current SoCs are still an important part of the total manufacturing cost of the system-chips, and one can still see a considerable effort for its reduction. The main challenges of the SoC testing are described in Section 2.3, and solutions tackling some of these challenges are described in Chapter 3.

2.2 NoC-based SoC Design

Advances in the SoC design methods and tools aim at reducing the design complexity by automating some integration and synthesis steps, such as the co-simulation and co-synthesis of distinct cores descriptions. Recently, the concept of **design platforms** has been introduced to define a common set of architectural blocks over which the system is

built. The platform makes the reuse easier, since the main part of a family of applications is implemented once, but used several times. Thus, for each new application, only a small variation on the design is actually implemented.

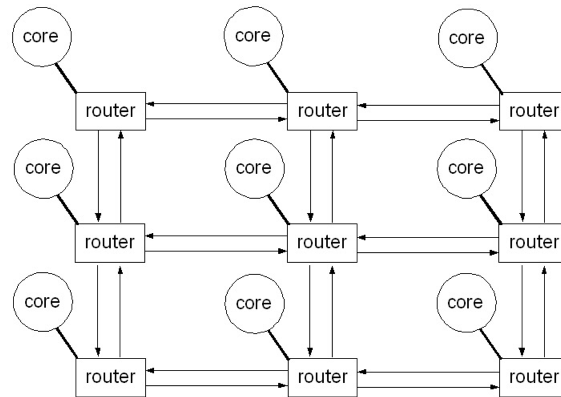
A platform is a common base of hardware and software components that can be reused for the design of a number of distinct systems (SANGIOVANNI-VINCENTELLI; MARTIN, 2001; KEUTZER et al., 2000). The hardware base can be composed by a micro-architecture almost fixed with one or more microprocessors and other peripheral components connected through a communication structure. For example, the common architectures typically include a CPU and memories communicating over a fast bus and peripherals communicating through a slow bus (BERGAMASCHI; COHN, 2002). As for the software, the base components can be, for example, a real-time operating system (RTOS) accessible through Application Programming Interface (API) routines. These common components are standardized in such a way that they do not need to be validated for each new project, to accelerate the design time. On the other hand, the platform must offer parameterization and configuration capabilities to be easily reusable.

In terms of communication capabilities of a SoC, future systems will probably require communication templates with several dozens of Gbits/s of bandwidth (cell phones, network applications, etc). Buses can not always fulfill the performance requirements of such systems without posing new problems to power consumption and design reuse. Thus, recent works (GUERRIER; GREINER, 2000; BENINI; MICHELI, 2002; DALLY; TOWLES, 2001) have proposed the use of a pre-defined platform to implement the communication among the several cores in a chip. Such a platform is implemented as an integrated switching network, called Network-on-Chip (NoC), and meets some of the key requirements of future systems: reusability, scalable bandwidth, and low power consumption.

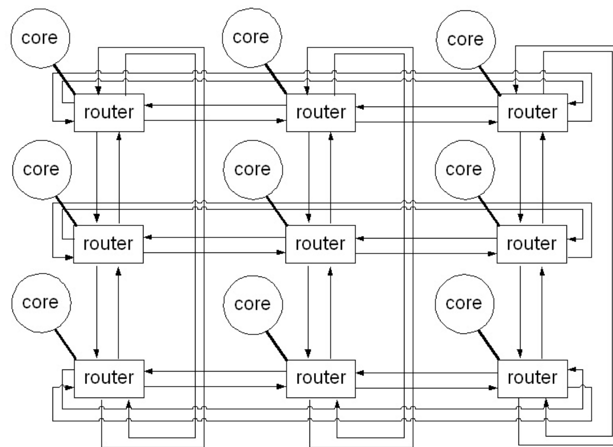
A study presented in (ZEFERINO et al., 2002) shows that NoCs have better communication performance than buses for a number as low as eight cores, if intensive communication, e.g. each core exchanging messages with another one, is required. For lighter workloads (fewer messages with reduced size), the performance of a central bus will be better than the NoC in systems with up to sixteen cores. Therefore, it is clear that NoCs can potentially become the preferred SoC interconnection approach in the near future.

Networks-on-Chip are based on the interconnection networks largely used in parallel computers. NoCs can be defined as a structured set of routers and point-to-point channels interconnecting the processing cores of a SoC in order to support communication among them. Such a structure can be described as a graph with routers on the nodes and channels on the arcs, and it is named topology. Some examples of topologies include grid, torus, hypercube, ring, multi-stage and fat-tree (DUATO; YALAMANCHILI; NI, 1997). In current NoCs, the preferred topologies are the ones with planar structures, because they are easier to be implemented with current technologies (BABB et al., 1999). Figure 2.3 shows three examples of NoC topologies: grid (Figure 2.3(a)), torus (Figure 2.3(b)), and fat-tree (Figure 2.3(c)).

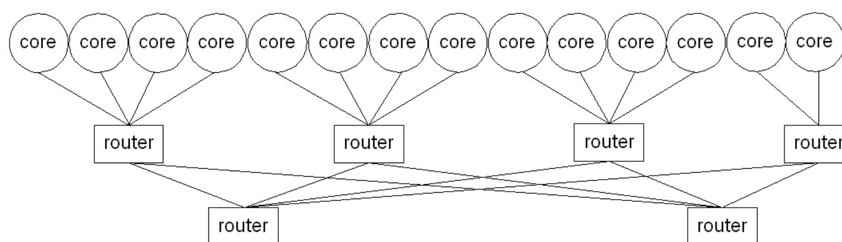
NoCs typically use the message-passing communication model, and the processing cores attached to the network communicate by sending and receiving request and response messages. A message forwards from a sender to a receiver by requesting and reserving resources of the network in order to establish a route between the sender and the receiver. To be routed by the network, a message is composed by a header, a payload and a trailer. The header and the trailer frame the packet and the payload carries the data being transferred. The header also carries the information needed to establish the path



(a) grid



(b) torus



(c) fat-tree

Figure 2.3: Some NoC topologies

between the sender and the receiver, and the trailer can be either an additional word in the message or a bit in the last word of the payload. Depending on the network implementation, messages can be split into smaller structures named packets, which have the same format of a message and are individually routed. Packet-based networks present a better resource utilization, because packets are shorter and reserve a smaller number of channels during their transfer. Besides its topology, a NoC can be described by the approaches used to implement the mechanisms for flow-control, routing, arbitration, switching and buffering, as follows. The flow control deals with data traffic on the channels and inside the routers. Routing is the mechanism that defines the path a message takes from a sender to a receiver. The arbitration establishes priority rules when two or more messages request the same resource. Switching is the mechanism that takes an incoming message of a router and puts it in an output port of the router. Finally, buffering is the strategy used to store messages when a requested output channel is busy. Current cores usually need to use wrappers to adapt their interfaces and protocols to the ones of the target NoC. Such wrappers pack and unpack data exchanged by the processing cores which.

Some implementations of such integrated networks can be found in (KARIM; NGUYEN; DEY, 2002), (FORSELL, 2002), and (ZEFERINO, 2003), for example.

2.3 Test Challenges in SoC Design

Figure 2.4, extracted from (MARINISSEN; ZORIAN, 1999), shows the main differences between the test of a system-on-board and a system-on-chip. In the SoB test, each integrated circuit (IC) mounted in the board is totally designed, manufactured and tested before becoming part of a more complex system. In a core-based SoC, on the other hand, all cores are tested together, after the whole system is synthesized and manufactured(ZORIAN, 1997). Although each core is assumed to be functionally correct, its behavior after manufacturing and in conjunction with other cores is not known a priori (GUPTA; ZORIAN, 1997).

Another key difference between the SoB and the SoC testing is the access to the cores periphery during test (ZORIAN; MARINISSEN; DEY, 1998). In a SoB, probes can normally be used to access each IC. Alternatively, boundary-scan chains are used to serially control and observe each block. As the number of ICs in the board is relatively small, the time required to scan-in and scan-out the test data is usually acceptable. In a SoC, probes can not be used to access the cores that are deeply embedded into the chip. Moreover, the use of boundary-scan to access internal blocks may be too costly in terms of time, as the number of cores in the system increases every day.

Furthermore, one of the main challenges of the SoC testing is the integration and coordination of the test and diagnose techniques of all cores that compose the circuit (ZORIAN, 1997). Whereas the test of a SoB consists basically on the test of the interconnections among ICs, in the SoC the system testing comprises not only the interconnection test, but also the verification of each core and the user defined logic as well.

One can divide the test requirements of the SoC in three levels (ZORIAN, 1998): core requirements, interconnection requirements, and system requirements.

2.3.1 Core Test Requirements

- **Definition of the core test approach**

  The definition of a test strategy for a core depends on the knowledge of the logic implemented by that block. Therefore, this task is usually performed by the core

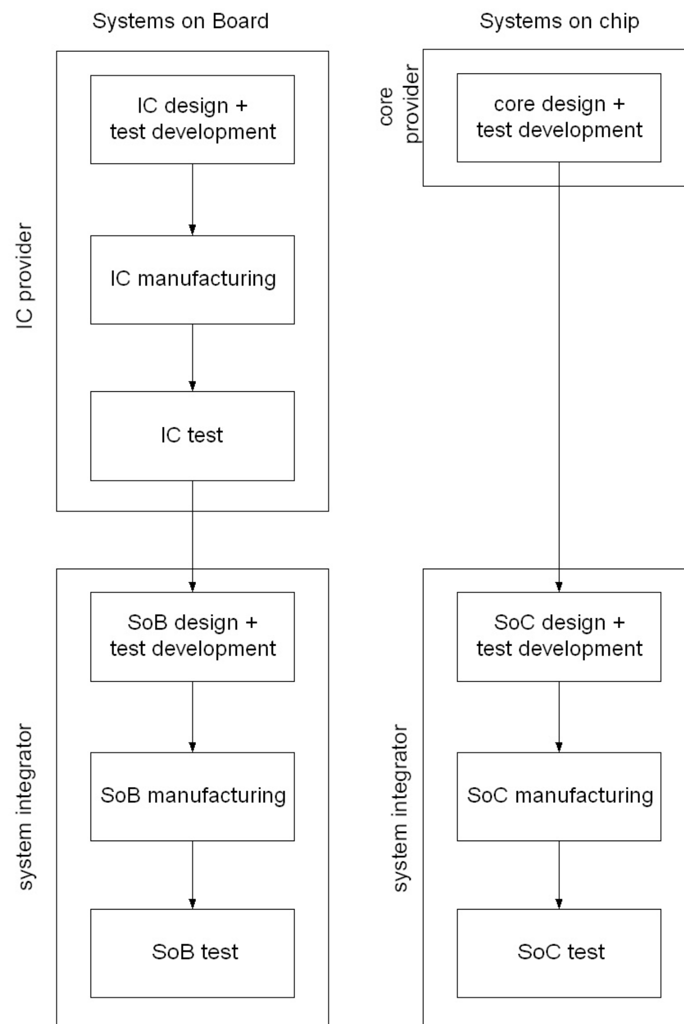


Figure 2.4: Differences between SoB and SoC testing (MARINISSEN; ZORIAN, 1999)

provider, which also assures the protection of the intellectual property associated to the reusable block. However, the core test strategy also depends on the target technology of the final system, the system test resources and the required fault coverage, but these parameters are not known by the core provider a priori. Thus, in general, the core provider supplies a basic set of test vectors and DFT strategies (scan chains, BIST controllers) for the core, to test for the most common technology faults. For open source soft cores, the system integrator has access to the core description. In this case, the integrator can make some modifications in the core logic, although this may require additional design time. When the core can not be modified by the integrator and more DFT structures or test patterns are required, an additional agreement between the core designer and the core user is necessary. In both cases, the communication between the two parts (core provider and core user) is the key for the successful testing of the block. The transmission of the core test strategy (test patterns, expected responses, control signals, etc) must be clear and unambiguous. On the other hand, if the system integrator needs other test schemes implemented in the core, he/she must specify very clearly which modifications must be implemented. Thus, although there are numerous test techniques for all types of logics (microprocessors, analog filters, DSP-based logics, among others) that can

be reused for the test of a core, the problem at this level is the information flow from the core provider to the core user and vice-versa.

- **Access to the core periphery during test**

  During test, other pins, in addition to the functional interfaces of the core must be accessed (scan-in and scan-out interfaces, control pins, testing clock, etc). As many cores are deeply embedded into the chip and the number of pins at system level is usually much smaller than the number of pins of a core, the transmission of the test data between an external tester and the embedded module is an important issue. Moreover, the definition of the access mechanism for each core impacts all other system test costs, such as test time, and area overhead. Therefore, the definition of such mechanism must be carefully considered.

- **Core isolation**

  During the test of a core, it is usually necessary to put this block in a test mode, so that the test pins become ready to receive and send data. Additionally, it may be necessary to isolate this block from the rest of the system so that other blocks are not damaged or can be tested in parallel. Therefore, the inclusion of an extra logic around the core to provide the several operation modes of this module is usually required. This logic can be either part of the core and be delivered along with the block itself, or be implemented by the system integrator, according to the core and system requirements (ZORIAN, 1997, 1998).

2.3.2 Interconnection Requirements

The test of the interconnections, in the second level, presents a single requirement: the possibility of precisely controlling and observing each connection. This test is, nevertheless, of extreme importance for the system characterization, since it can determine the actual performance achieved by the system. Furthermore, as the number of connections may be high, it has an important impact on the system test time. This test must be defined by the system integrator. In general, the interconnection test relies on the existence of some mechanism around the core (a boundary register, for example) that allows the load and capture of the interconnection signals.

2.3.3 System Requirements

- **Test of the UDL**

  Some authors consider the interconnections as part of the user defined logic. However, this logic can also implement some other functions, such as the data conversion between two cores. In the first case, the UDL testing is treated as the interconnection testing. In the second case, the UDL can be viewed as another core in the system, but that can be modified by the system integrator. Thus, the test of this logic can be defined as the test of an additional soft core.

- **Test scheduling definition**

  The test scheduling defines the order of testing of each part of the system: cores, interconnections and UDL. The scheduling depends basically on the set of test resources available and shared among cores, and on the system power constraints.

- **Test controller**

  The test controller is the module that runs the test program, sending the correct control and test signals to each part under test in the system. This controller can be an automatic test equipment, outside the chip, or can be implemented inside the chip.

- **Test integration**

  The combination of the access mechanism of each core in such a way that all cores are properly tested without deeply affecting the system performance, cost and design time is the most important system test requirement. Actually, this system-level requirement is the combination of all requirements previously defined, and represents the complexity of the SoC testing faced by the system integrator. For example, the system integrator has to verify that the access mechanisms defined for the cores can be implemented and synthesized with the system logic, without compromising the application performance and the chip cost. Moreover, the integrator has to define a test scheduling of minimum time while still meeting the application power constraints. In fact, the test integration is very similar to the system integration, and one can certainly agree that there are several possible test solutions, considering different access mechanisms, different cores versions (BISTed and non-BISTed, for instance), and different synthesis possibilities, that meet all test requirements. Therefore, the design space considering the test is quite large. Moreover, because of this complexity, the system integrator must consider the possible test solutions as early as possible in the design flow.

According to the system design model, the test requirements can be more or less modified. For example, the flexibility of the soft cores can be used to facilitate the combination of the cores test requirements and the system constraints. Bus-based functional connections are probably faster to test, if the connections are centralized. NoC-based designs, on the other hand, may have more connections to be tested, but the communication protocol and the possibility of reuse of the communication platform may accelerate the test definition.

In Chapter 3, a summary of the solutions that have been presented in the last few years, in response to some of the defined test requirements, is presented. Then, these solutions will be discussed and the motivation and intended contributions of this work will be further detailed.

3 RELATED WORKS AND MOTIVATIONS

A great deal of effort has been expended in the last few years, towards the development of suitable solutions for the test of core-based systems. Responding to the several test requirements listed in Chapter 2, one can group the techniques presented so far in four categories: test access mechanism definition, test scheduling methods, test planing approaches, and standardization initiatives. The techniques in the first group tackle the problem of defining an access mechanism to the cores periphery during test. Such solutions can either be based on the reuse of system functional connections, cores logic or on the insertion of a test bus. In the second group, the minimization of the test time is addressed, usually based on an access mechanism previously defined. In the third group one will find the techniques that take into account a number of system or test aspects. Finally, the fourth group comprises the initiatives for the standardization of the interface between the cores and the system during test, and the definition of a set of SoC test benchmarks. In the sequel, some representative works of these four groups will be discussed and the remaining test requirements, still not tackled by the available solutions, will be listed.

3.1 Test Access Mechanism Definition

Zorian *et al.* (ZORIAN; MARINISSEN; DEY, 1998) introduced the generic conceptual test access architecture for embedded cores as well as a nomenclature for its elements that has been used in the literature. The conceptual architecture, shown in Figure 3.1, is composed of four basic elements:

- a test *stimuli* source for the real-time test pattern generation;
- a test sink for the reception and evaluation of the test responses;
- a Test Access Mechanism (TAM) for the transportation of the test data from the test source to the core and from the core to the test sink;
- a core wrapper, for the connection of the core terminals to the TAM terminals, providing the mechanisms for isolation and integration of the core to the system during test.

These four elements can be implemented in several ways, according to the core requirements and system constraints. However, the wrapper structure must allow the core to operate in at least three modes: normal, internal test, and external or interconnection test. Additionally, the wrapper must also implement some type of bypass mode to isolate the core from the system when other cores are being tested.

The test sources and sinks can be implemented in a number of ways:

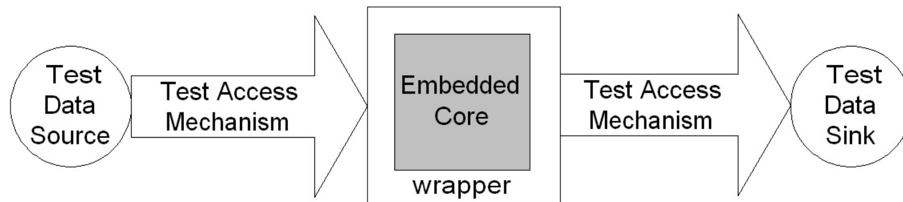


Figure 3.1: Conceptual architecture for the SoC testing (ZORIAN; MARINISSEN; DEY, 1998)

- off-chip, using an external test equipment;
- on-chip, through BIST structures;
- as a combination of both (for example, when a core is tested by a combination of deterministic and pseudo-random vectors).

In addition, the source and the sink do not need to be of the same type, that is, one can have an on-chip test source and an off-chip test sink, or vice-versa. As one will see in the several test approaches presented in the following, the choice for one implementation of a test source, sink or access mechanism depends on the type of cores embedded into the system, on the test requirements of such cores, and on a number of system constraints, such as area, test time, time-to-market, and so on. For example, on-chip sources and sinks usually present a better fault coverage than their off-chip version, but may increase the system area, leading to yield reduction. On the other hand, off-chip sources and sinks usually require more elaborated access mechanisms and may increase the system test time.

The test wrapper is a thin shell around the core that connects the TAM(s) to the core (MARINISSEN; KAPUR; ZORIAN, 2000). The wrapper provides the switching between normal functional access and test access via the TAM. Well designed wrappers provide test access for both core-internal testing as well as core-external testing. Furthermore, wrappers may provide width adaptation in case of a mismatch between core I/O width and TAM width (MARINISSEN; KAPUR; ZORIAN, 2000).

The test access mechanism communicates the core under test to the pattern sources and the test sinks. Although the same mechanism can be used for the transport of the test data in both directions, this is not mandatory and a number of TAM combinations can co-exist for the same core (ZORIAN; MARINISSEN; DEY, 1998). The design of a TAM always searches the best trade-off between the transport capacity of the mechanism and its application cost. The capacity o f data transportation is limited by the capacity of the source and sink, and by the system area that can be used by the TAM, which is usually measured as the TAM bitwidth. Figure 3.2 shows the relationship between the TAM bitwidth and the test costs in terms of area and pin overhead, and the core test time.

Zorian *et al.* (ZORIAN; MARINISSEN; DEY, 1998) list the following options for the TAM definition:

- the reuse of functional buses and interconnections of the system;
- the use of specific test access inserted into the system;
- the reuse of the cores or other logic blocks for the test access path, either using the cores functionality (called transparent modes) or some type of bypass mechanism ;

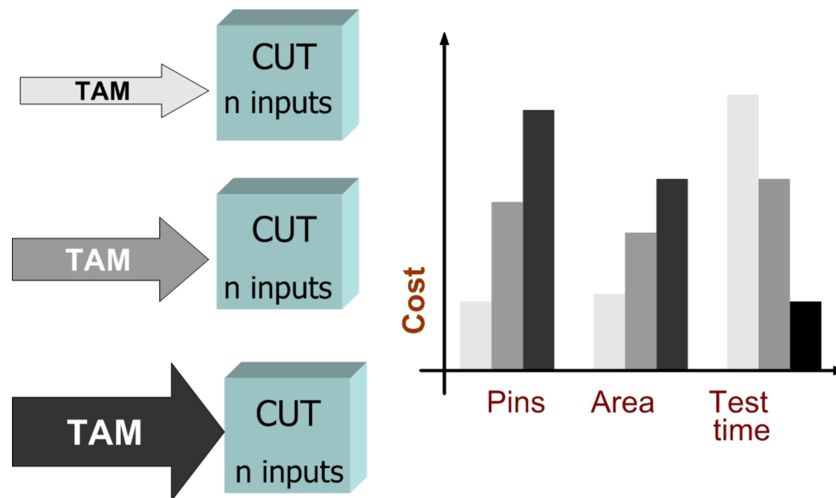


Figure 3.2: Relationship between TAM width and test costs

- the use of an independent access for each core or the TAM sharing among cores;
- The transportation of only the test data by the TAM or the inclusion of some control functions in the access mechanism itself.

In the following, the most important approaches for TAM definition presented so far are described. These approaches range from adaptations of traditional test techniques, such as the 1149.1 Standard (IEEE Standards Board, 1990), to the development of hierarchical and scalable methods for test access. As for the performance of the test solution, the proposed approaches create the TAM mainly considering testing time issues.

(WHETSEL, 1997), (BHATTACHARYA, 1998), (LEE; HUANG, 2000), (HU; YIBE, 2001), (LI et al., 2002a), (OAKLAND, 2000), and (LI et al., 2002b) propose test access mechanisms based on the IEEE 1149.1 boundary-scan standard, also known as JTAG standard (IEEE Standards Board, 1990). The assumption of these approaches is that many cores being used today were ASICs in the past, and the boundary-scan is already implemented for those modules. Moreover, as the JTAG mechanism requires only five extra pins at system-level, the test cost is drastically reduced. However, the inclusion of a TAP (Test Access Port) controller into a core makes the integration of such a core into a SoC (LOUSBERG, 2002) more difficult, since an extra level of controlling is required for each TAPed module. The main disadvantage of the JTAG-based methods is, nevertheless, the possibly excessive testing time caused by the reduced TAM bandwidth provided by the system-level TAP.

Other authors tackle the access problem by reusing available system resources, such as cores and functional interconnections.

Ghosh *et al.* propose a method in which test access to embedded cores is based on transparent paths through other cores and design modules (GHOSH; JHA; DEY, 1997). In the proposed method, every core should not only come with a set of pre-computed tests, but also with a set of transparent paths, capable of transporting test data through the core. If these paths are originally not available, the core provider should add design-for-test hardware to the core in order to create test access paths to other cores. Some methods for the synthesis of transparent cores are proposed in (CHAKRABARTY; MUKHERJEE; A., 2001), (MAKRIS; ORAILOGLU, 1998), and (YONEDA; FUJIWARA, 2002). However, the transparent-based access method does not seem to address the issue of time-to-market

and in many cases yields an excessive number of access paths. In (GHOSH; DEY; JHA, 1998), this problem is considered and different transparent paths are available in different versions of each core, each version with a distinct area overhead so that only one version of the core is chosen as the system area is optimized. However, if the transparent modes are defined previously to the system integration, one can not assure that all required paths will be available or a huge number of core versions is still required, and the time-to-market issue can still be a problem. On the other hand, to define the transparent modes during the system integration, soft cores are assumed, and the access to the core description is required, which is not always the case in current SoCs. Chiusano *et al.* propose in (CHIUSANO; PRINETTO; WUNDERLICH, 2000) the use of arithmetic cores to generate test patterns for subsequent cores. In this case, the core does not implement a transparent mode, but its original logic is used as a pseudo-random test pattern generator.

Nourani and Papachristou propose in (NOURANI; PAPACHRISTOU, 1998a,b) and in (NOURANI; PAPACHRISTOU, 1999) the definition of the test access architecture by taking advantage of the connections already present in the system. A *bypass* mode is introduced for each core input port to its output port through which the test data can be transferred. The system is modeled as a directed weighted graph in which the core accessibility is solved as a shortest path problem. This model has been improved and presented in (NOURANI; PAPACHRISTOU, 2000), where other structures in the system, such as buses and tri-state ports, are also considered for the test path definition. The problem of TAM definition and test time minimization is formulated and solved as an ILP problem.

The reuse of the microprocessor that is usually present in the system is proposed in a number of works (PAPACHRISTOU; MARTIN; NOURANI, 1999), (HWANG; ABRAHAM, 2001), (LAHIRI; RAGHUNATHAN; DEY, 2002), (CHEN; BAI; DEY, 2002). The basic assumption of those methods is that the microprocessor is connected to a large number (if not all) embedded cores through a functional bus or a hierarchy of buses. If this is not the case, additional hardware is inserted to make it possible the access and control of the core under test by the microprocessor. However, as there is a single or a small number of test processors being used, only one or a few cores are tested at a time. Therefore, this type of solution is usually effective for small systems where the microprocessor of the application is connected to most embedded cores. For those systems, the test time is affordable, as well as the area overhead caused by the access mechanism. On the other hand, if deterministic patterns are used, an external memory may still be required and the reduced pin count at the system interface may be a problem. If internal memories are also reused, in addition to the microprocessor reuse, they may be not large enough for the storage of all test patterns, and a mechanism for the memory reload is required.

Finally, a third line of solutions defends the insertion of additional buses in the system, called test buses, as the access mechanism. Despite the fact that the dedicated wiring increases the area costs of the SoC, the scalability and the possibility of modeling the problem with a limited number of variables, makes this TAM a very interesting and the most explored test architecture.

Varma and Bhatia describe a test access mechanism for embedded cores, named VisibleCores (VARMA; BHATIA, 1998). Their approach is based on two dedicated on-chip variable-width buses, one for transporting test control signals, and one for transporting test data signals. Embedded cores can be either connected or disconnected from the test buses. Test access from chip pins to embedded cores and vice versa is achieved by connecting the core-under-test to the test data bus, and disconnecting all other cores. A disadvantage

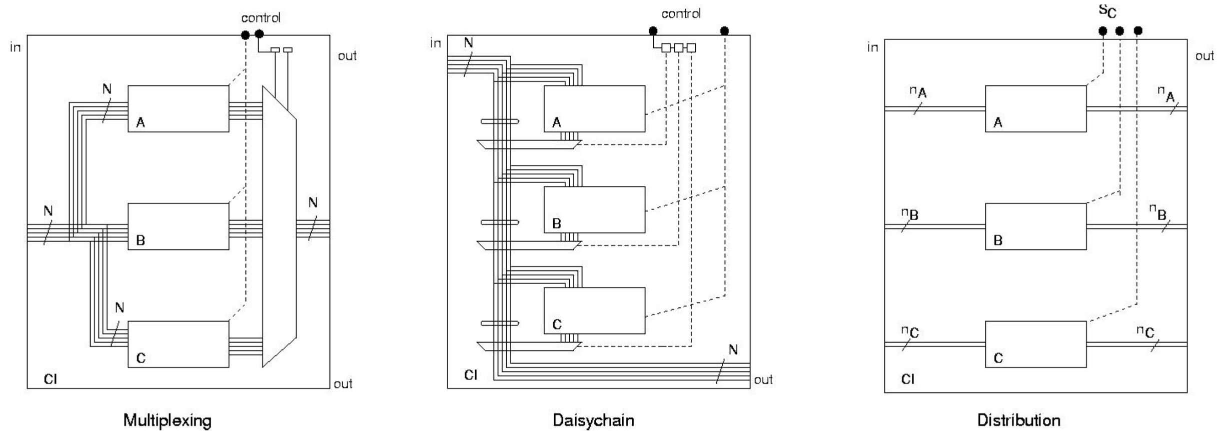


Figure 3.3: Test Bus Architectures (AERTS; MARINISSEN, 1998)

of this method is that only one core at a time can be connected to the test bus, while some tests involve multiple cores.

Three basic types of scalable TAMs have been described in (AERTS; MARINISSEN, 1998) and are exemplified in Figure 3.3: (a) the Multiplexing architecture, (b) the Daisychain architecture, and (c) the Distribution architecture.

In the Multiplexing and Daisychain architectures, all cores have access to the total available TAM width, while in the Distribution architecture, the total available TAM width is distributed over the cores. Note that the multiplexer in the Multiplexing Architecture is conceptual, and hence could also be implemented by means of tri-state buffers with appropriate control signals. In the Multiplexing architecture, only one core wrapper can be accessed at a time. Consequently, in this architecture the cores must be serially tested. An even more serious drawback of this architecture is that testing the circuitry and wiring in between cores is difficult; interconnect test requires simultaneous access to multiple wrappers. The other two basic architectures do not have these restrictions; they allow for both serial as well as parallel test schedules, and also support interconnect testing.

The TestRail architecture proposed in (MARINISSEN et al., 1998) and shown in Figure 3.4 is a combination of the Daisychain and Distribution architectures. A single TestRail is basically the same as what is described by the Daisychain architecture: scan-testable cores connected to the same TestRail can be tested simultaneously, as well as sequentially. A TestRail architecture allows for multiple TestRails on one SoC, which operate independently, as in the Distribution architecture. The TestRail architecture supports serial and parallel test schedules, as well as hybrid combinations of those.

In most test access architectures, the cores assigned to a TAM are connected to all wires of that TAM. Such architectures are referred to in (IYENGAR; CHAKRABARTY; MARINISSEN, 2002a) as fixed-width TAMs. The flexible-width TAMs, on the other hand, refer to the core-TAM assignments where the granularity of TAM wires are considered, instead of considering the entire TAM bundle as one inseparable entity. Figure 3.5 shows an example of a flexible-width Test Bus architecture.

In (CHAKRABARTY, 2000a), Chakrabarty proves that many problems related to the definition of bus-based TAMs are NP-hard. Additionally, he uses the Integer Linear Programming (ILP) heuristic to model and solve the test bus assignment problem. In this problem, for a given number of test pins at the system interface divided into a given number of test buses, the best width for each test bus and the best assignment of test buses to

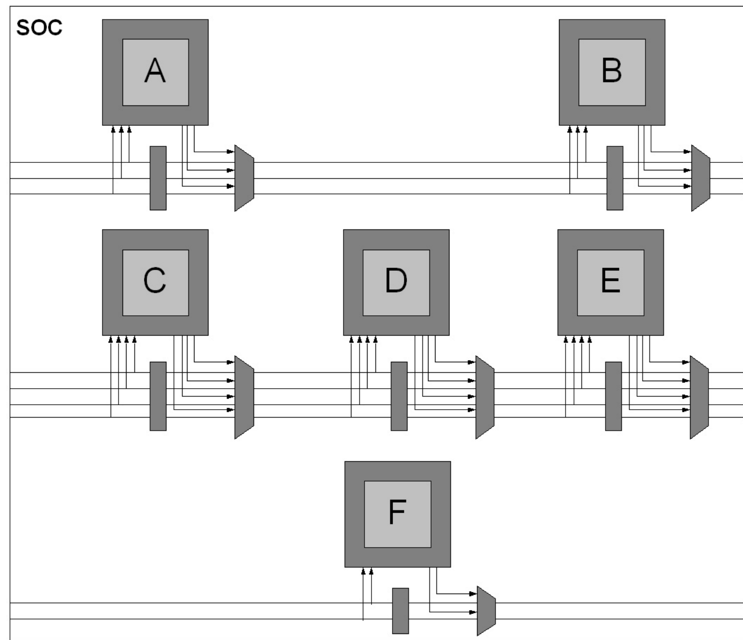


Figure 3.4: Test Rail Achitecture (MARINISSEN et al., 1998)

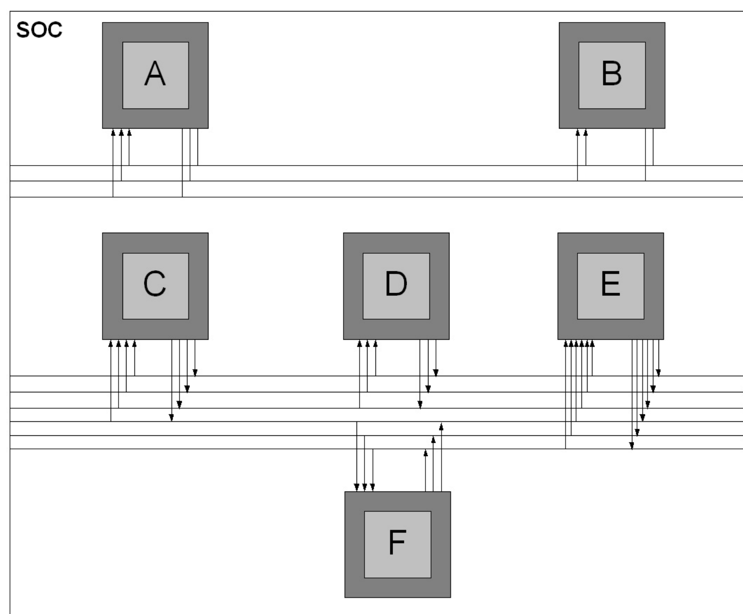


Figure 3.5: Flexible-width test bus architecture (IYENGAR; CHAKRABARTY; MARINISSEN, 2002a)

cores is defined so that the system test time is minimized. In further works, Chakrabarty, Iyengar, Marinissen, and others have improved this original ILP model to optimize the test bus assignment under other system constraints: power and place&route constraints are considered in (CHAKRABARTY, 2000b); wrapper and TAM co-optimization are addressed in (IYENGAR; CHAKRABARTY, 2001; IYENGAR; CHAKRABARTY; MARINISSEN, 2001) and (IYENGAR; CHAKRABARTY; MARINISSEN, 2002b) using new mathematical models that improve both the final solution and the execution time of the original ILP model. Test data compression for data volume and access requirements reduction is discussed in (IYENGAR et al., 2003).

Fixed-width TestRail architecture optimization was investigated in (GOEL; MARINISSEN, 2002a,b). These papers describe heuristic algorithms for co-optimization of wrappers and TestRails. In (GOEL; MARINISSEN, 2002c), a novel architecture-independent heuristic algorithm that optimizes the test architecture for cores with both fixed-length and flexible-length scan chains is proposed. The algorithm efficiently determines the number of TAMs and their widths, the assignment of modules to TAMs, and the wrapper design per module.

Other authors propose alternative methods for the bus assignment problem, which usually present better execution times if compared to the ILP model for similar system testing times: Ivanov *et al.* (EBADI; IVANOV, 2001), for example, use a genetic algorithm to solve the problem. In (HUANG et al., 2001), Test Bus architecture optimization is mapped to the problem of two-dimensional bin packing and a Best Fit algorithm is used to solve it.

The issue of designing balanced scan chains within the wrapper was addressed in (CHAKRABORTY; BHAWMIK; CHIANG, 2000). The first techniques to optimize wrappers for test time reduction were presented in (MARINISSEN; KAPUR; ZORIAN, 2000). To solve the problem, the authors proposed two polynomial-time algorithms that yield near-optimal results. Further, to perform wrapper optimization, Iyengar *et al.* proposed in (IYENGAR; CHAKRABARTY; MARINISSEN, 2001) and in (IYENGAR; CHAKRABARTY; MARINISSEN, 2002c) an algorithm based on the Best Fit Decreasing heuristic for the Bin Packing problem. The algorithm has two priorities: (i) minimizing core testing time , and (ii) minimizing the TAM width required for the test wrapper. These priorities are achieved by balancing the lengths of the wrapper scan chains originally designed, and identifying the number of wrapper scan chains that actually need to be created to minimize testing time.

The use of crossbar switches for the efficient communication at varying bitwidth between cores and the test bus is proposed in (BENABDENBI; MAROUFI; MARZOUKI, 2000, 2002) and (BASU et al., 2002,?).

Nahvi and Ivanov propose in (NAHVI; IVANOV, 2001) the use of a packet switching communication-based TAM for a SoC. The TAM proposed model is called NIMA (Novel Indirect and Modular Architecture), and it is defined to allow modularity, generality, and configurability for the test architecture. Such a architecture is very similar to a functional on-chip network , but it is specifically designed for the test task. Thus, routing and addressing strategies are defined considering the test requirements of each system. For example, the messages containing test responses do not present a target address, for they are scheduled by the test sink. Moreover, routing is hardwired, assuming that a test schedule is defined by the system designer before the system synthesis. The results presented in that work show the good performance of this TAM model with respect to area overhead and test time, when compared to bus-based TAMs. However, the extra area required for

the network implementation can be prohibitive.

3.2 Test Scheduling Definition

Sugihara *et al.* (SUGIHARA; DATE; YASUURA, 1998, 2000) and Jervan *et al.* (JERVAN; PENG; UBAR, 2000; JERVAN et al., 2002) propose automatic methods to select the best combination of pseudo-random and deterministic patterns for each core in an SoC to minimize the system test time. In both works, each core is assumed to have a number of test sets, each set using a different combination of pseudo-random and deterministic patterns. The selection algorithms operate over the defined test sets, choosing one set for each core so that the system test time is minimized. In (SUGIHARA; DATE; YASUURA, 2000) the usage of BIST controllers and TAM is also optimized. The quality of the final solution for these approaches is related to the number of possible test sets, which defines the search space for the selection algorithms. In addition, the test time minimization is based on the test parallelization of the BISTed cores.

In (ZHAO; UPADHYAYA, 2002) Zhao and Upadhyaya consider a system where one test set or a combination of test sets may be provided for testing each core in order to provide the required fault coverage. Given a set of test sets for the cores, a set of resources, the test access architecture and the maximum power allowance, they propose a test scheduling scheme to minimize the overall test time by efficiently overlapping blocks of compatible tests of unequal length. The basic idea is to generate a group of power-constrained concurrent test sets (PCTS) and schedule the tests based on the compatibility relations among them. The test scheduling dynamically partitions and allocates the tests, consequently constructing and updating a set of dynamically partitioned power constrained concurrent test sets, and ultimately reducing the test application time.

Integer Linear Programming was also used to solve the test scheduling problem. In (CHAKRABARTY, 1999, 2000c), the problem is modeled as a m-processor open-shop scheduling which is NP-hard. Then a mixed ILP (MILP) heuristic for optimal scheduling and optimal test set selection is used. Additionally, another heuristic is proposed to handle larger systems for which the MILP model may be unfeasible.

SoCs in test mode can dissipate up to twice the amount of power they do in normal mode, since cores that do not normally operate in parallel may be tested concurrently (ZORIAN, 1993). Power-constrained test scheduling is therefore essential in order to limit the amount of concurrency during test application to ensure that the maximum power budget of the SoC is not exceeded (IYENGAR; CHAKRABARTY; MARINISSEN, 2002a). In (CHOU; SALUJA; AGRAWAL, 1997), a method based on approximate vertex cover of a resource-constrained test compatibility graph was presented. In (MURESAN; WANG; VLADUTIU, 2000), the use of list scheduling and tree-growing algorithms for power-constrained scheduling is discussed. The authors presented a greedy algorithm to overlay tests such that the power constraint is not violated.

Ravikumar *et al.* (RAVIKUMAR; VERMA; CHANDRA, 1999) present a polynomial-time algorithm for finding an optimum power-constrained schedule which minimizes the test time. They assume the built-in self-test (BIST) methodology for testing individual cores and allow sharing of test resources (pattern generators and signature registers) among cores. The objective is to minimize the test application time and the test area overhead, treating the total power dissipation as a constraint. Further, they expand their work in (CHANDRA, 2000) and propose an algorithm for simultaneous module selection and test scheduling under power constraints. The method relies on the existence of a li-

brary of possible mappings for each core in the system, each one with a different power consumption and area. The algorithm thus selects which version of each core should be synthesized so that a minimal test time for a given power constraint is found.

Larsson and Peng present an integrated technique for test scheduling and scan-chain division under power constraints in (LARSSON; PENG, 2001a). The authors presented an optimal algorithm to parallelize tests under power and resource constraints. The design of test wrappers to allow for multiple scan chain configurations within a core was also studied. Larsson and Fujiwara extended this model in (LARSSON; FUJIWARA, 2002) by (1) allowing several different bandwidths at cores and (2) controlling the cores test power consumption, which makes the increase of the test clock possible. The scheduling is modeled as a Bin-packing problem and the transformations of TAM-time and power-time, and the possibilities to achieve an optimal solution are discussed.

In (ROSINGER; AL-HASHIMI; NICOLICI, 2001, 2002), Rosinger *et al* propose a power profile manipulation approach for the minimization of the power dissipation during test. Such manipulation provides a more realistic power profile to be used by any power constrained test scheduling algorithm, making it possible the test time reduction by increasing the test concurrency.

In (IYENGAR, 2001; IYENGAR; CHAKRABARTY, 2002), an integrated approach to test scheduling is presented. For precedence-based scheduling of large SoCs, a heuristic algorithm was developed. The proposed approach also includes an algorithm to obtain preemptive test schedules in $O(n^3)$ time, where n is the number of tests (IYENGAR; CHAKRABARTY, 2002). Parameters that allow only a certain number of preemptions per test can be used to prevent excessive BIST and sequential circuit test preemptions. Finally, a new power-constrained scheduling technique was presented.

Pouget *et al.* present in (FLOTTES; POUGET; ROUZEYRE, 2002) a sessionless test scheme. Several constraints in terms of test resource sharing, power dissipation and precedence are taken into account. The problem is solved using a $O(n^3)$ heuristic, for n the number of cores in the system.

In (POMERANZ; REDDY, 2002) the core-clustering strategy is used to achieve optimal test completion time for the SoC. The method takes advantage of the existence of more than one copy of a core in an SoC to reduce test application time. This is achieved by the use of core clustering which is simultaneously considered with wrapper design, pin association, and the system power constraints.

Both TAM optimization and test scheduling significantly influence the test time, test data volume and test cost for SoCs. Furthermore, TAMs and test schedules are closely related. For example, an effective schedule developed for a particular TAM architecture may be inefficient or even unfeasible for a different TAM architecture. Integrated methods that perform TAM design and test scheduling in conjunction are therefore required to achieve low-cost, high-quality tests.

In (HUANG et al., 2002), Huang *et al.* present a method to solve the resource allocation and test scheduling problems together in order to achieve concurrent test for core-based SoC designs. The proposed methodology is not limited to any specific TAM and the problem is formulated as a 2-dimensional bin-packing problem. A best-fit heuristic algorithm is adopted to achieve optimal solution.

Koranne formulates the test scheduling as a network transportation problem in (KORANNE; CHOUDHARY, 2002; KORANNE, 2002). Given a set of tests, with demands for transportation of test bits (either for test stimuli or test response) and unrelated parallel test resources (e.g., test access mechanisms or builtŮin self-test engines), the method

determines the start times and resource mappings of all the tests such that the finish time for the complete SoC test is minimized. The problem is NP-hard and an approximation algorithm using a result from the solution of the single source unsplittable flow problem is presented. The proposed method uses the number of test bits that need to be transported for a test as the invariant and is hence relatively independent of the test application and execution model.

Koranne and Iyengar propose in (KORANNE; IYENGAR, 2002) the representation of SoC test schedules and TAM width assignment based on the use of k-tuples, providing a compact and standardized representation of the test schedules and facilitating a fast and efficient evaluation of SoC test automation solutions. They further propose the use of heuristic algorithms based on the use of k-tuples to solve scheduling problems considering precedence relations among tests and power constraints.

Finally, genetic algorithms have also been used by Chattopadhyay and Reddy to solve the problems of test scheduling and TAM partition for systems-on-chip (CHATTOPADHYAY S.; REDDY, 2003).

3.3 Test Planning

Recently, a number of frameworks and increasingly more comprehensive models have been proposed to cope with global optimizations for the final solution. These frameworks differ by the type of core test methods addressed and the TAM definition, but they all manage the distribution of test resources considering a variety of cost factors.

Benso *et al* present in (BENSO et al., 2000) a tool for integration of cores with different test requirements (full scan, partial scan and BIST ready cores). The TAM follows a bus-based model that connects a group of cores to a BIST controller. Scheduling of BIST resources and data pattern delivery are also considered in the test solution.

Larsson and Peng propose in (LARSSON; PENG, 2001b; LARSSON; PENG; CARLSSON, 2001; LARSSON et al., 2002; LARSSON; PENG, 2002) a framework for SoC testing which considers test time minimization, TAM optimization, test set selection and test resource placement, along with test resources and power consumption constraints. The tool is based on the fact that different test sets can be used to test a core. This way, each test set is evaluated under power, time, memory requirements, and so on, and the best test set is chosen according to the system constraints. They further assume that scan chains can be divided into smaller ones, to accelerate test time. BIST resources are then placed in the system according to their usage by the cores. After the TAM definition, the test schedule is generated so that the minimum test time for that specific TAM is achieved.

Iyengar *et al.* describes in (IYENGAR; CHAKRABARTY; MARINISSEN, 2002d) an integrated framework for plug-and-play SoC test automation. The framework is based on a new approach for wrapper/TAM co-optimization based on rectangle packing. Additionally, the rectangle packing approach is used to develop an integrated scheduling algorithm that incorporates preemptive, precedence and power constraints in the test schedule. Moreover, the relationship between the TAM width and the tester data volume is studied to identify and effective TAM width for the SoC.

Data volume reduction has been addressed to reduce the memory requirements in the external tester and further reduce the system test time (TOUBA, 2002; SINANOGLU; ORAILOGLU, 2002; GONCIARI; AL-HASHIMI; NICOLICI, 2002; ROSINGER et al., 2001; CHANDRA; CHAKRABARTY, 2002a,b).

In (KUMAR GOEL; MARINISSEN, 2003), Goel and Marinissen extend existing SoC

test architecture design approaches to minimize the required tester vector memory depth and test application time, with the capability to minimize the wire length required by the test architecture. The user specifies the relative weight of the costs of test time versus wire length. In an integrated fashion, the algorithm partitions the total available TAM width over individual TAMs, assigns the modules to these TAMs, and orders the modules within one TAM such that the total cost is minimized.

3.4 Test Standard Initiatives

3.4.1 IEEE P1500 Standard

In September 1995, the IEEE Test Technology Technical Committee (TTTC) created a Technical Activity Committee (TAC) for the study of the test and design-for-test of core-based SoCs. This committee became the IEEE P1500 Standard for Embedded Core Test (SECT) group in June 1997, with the main goal of developing a standard mechanism for the test of core-based systems (MARINISSEN et al., 1999).

IEEE P1500 SECT is an IEEE standard under development that intends to facilitate core-based testing, i.e., testing embedded cores for modular testing of large system chips that consist entirely of modules of which not all implementation details are known. The motivation behind this industry-wide standard is to enable the reuse of tests when a core gets embedded in multiple different SoCs, as well as to enable inter-operable core-based testing of SoCs that contain multiple cores from distinct core providers. IEEE P1500 SECT standardizes a test information transfer model as well as (part of) the on-chip test access hardware that enables core-based testing (MARINISSEN et al., 2002).

IEEE P1500 does not cover the internal test methods of the cores or DfT, nor SoC test integration and optimization. These are completely in the hands of core providers or core users respectively, and are not suited for standardization because their requirements differ for the different technologies and design styles of different cores and SoCs (MARINISSEN et al., 2002).

The two main elements of the P1500 standard are a language, called Core Test Language (CTL) and a scalable core test architecture (MARINISSEN et al., 1999; HALES; MARINISSEN, 2003). The first one is developed by the P1500 CTL Task Force and meant to standardize the core test knowledge transfer. CTL is based on another IEEE standard language called Standard Test Interface Language or STIL, also known as IEEE 1450.0 (BOARD, 1999; ALLIANCE, 2003a). STIL is being extended to accommodate specific core test constructs (KAPUR et al., 2001).

3.4.1.1 The Core Test Description Language

The Core Test Description Language (CTL) focuses on defining a standard language in which all test-related information to be transferred from core providers to core users can be expressed (KAPUR et al., 1999). CTL is designed to work with cores that come with any type of DFT methodology. Furthermore, CTL also works with any type of test methodology or fault model (structural or functional test, delay fault model, Iddq tests, and so on) (KAPUR et al., 2001).

Information in CTL is organized around a mode (configuration) of the core being described. Using some basic constructs derived from STIL, CTL has the ability to describe the following (KAPUR et al., 2001):

- Controls to configure for testing the core and the surrounding SoC logic;

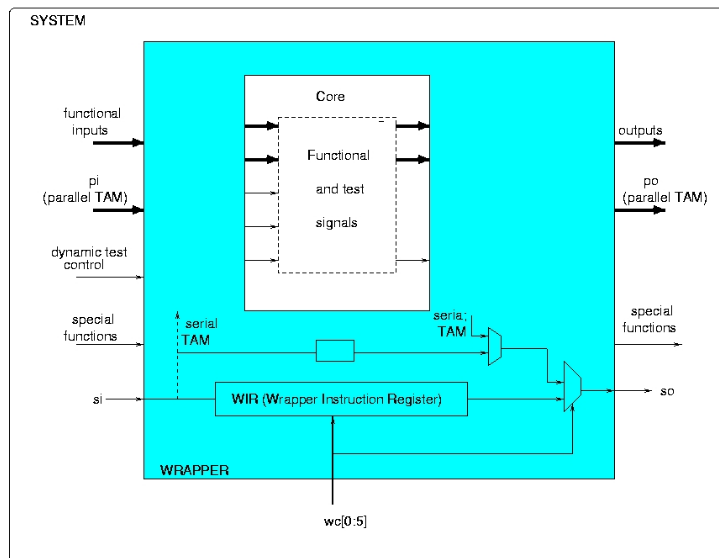


Figure 3.6: P1500 wrapper structure (MARINISSEN et al., 2002)

- Requirements and constraints on the implementation of SoC-level interfaces to the core;
- Inclusion of test data specific for the core, but defined independently from any particular use of the core.

3.4.1.2 Scalable Core Test Architecture

From the basic elements of the conceptual test architecture shown in Figure 3.1, IEEE P1500 SECT only standardizes the wrapper (KAPUR et al., 2001). The P1500 wrapper is a shell around a core, that allows that core to be tested as a stand-alone entity by isolating it from its environment. Likewise, the wrapper allows the environment to be tested independent from the state of the core. The wrapper has three main types of modes: (1) functional operation, in which the wrapper is transparent and operates as if not existing, (2) inward-facing test modes, in which test access is provided to the core itself, and (3) outward-facing test modes, in which test access is provided to the circuitry outside the core (MARINISSEN; GOEL; LOUSBERG, 2000). Figure 3.6 gives an overview of the main elements of the P1500 wrapper architecture (MARINISSEN et al., 2002). The P1500 wrapper is shown as a shell around the core. It has functional input and output ports, matching those of the unwrapped core. Furthermore it has a mandatory one-bit input/output port pair, WSI (*Wrapper Serial Input*) and WSO (*Wrapper Serial Output*), and optionally one or more multi-bit input/output port pairs; in Figure 5.4, one is drawn, named WPI (*Wrapper Parallel Input*) and WPO (*Wrapper Parallel Output*). The control of the operation modes is done by the Wrapper Instruction Register, that can be loaded either serially, through WSI, or in parallel.

IEEE P1500 SECT supports two compliance levels (MARINISSEN et al., 1999), commonly referred to as IEEE 1500 Unwrapped and IEEE 1500 Wrapped (KAPUR et al., 2001). In both cases, the core comes with a CTL program that describes the core tests. In the case of a 1500 Wrapped core, the core incorporates a complete P1500 wrapper function, while for a 1500 Unwrapped core, the wrapper is not present yet, but the CTL program contains the information on the basis of which a compliant wrapper can be

added (KAPUR et al., 2001). Although the benefits of modular testing, test interoperability, and test reuse only become apparent when indeed the P1500 wrapper is used, the two compliance levels provide flexibility in the usage of the standard. The first version of IEEE P1500 SECT focuses on non-merged digital logic and memory cores. This standard is currently in its development phase; the latest internal draft standard document (IEEE P1500/D0.5) has been released in October 2001 (HALES; MARINISSEN, 2003). After completion of this standard, P1500 also intends to cover analog and mixed-signal cores, as well as DfT guidelines for mergeable cores (MARINISSEN et al., 2002).

3.4.2 VSI Alliance

The Virtual Socket Interface Alliance (VSIA) (ALLIANCE, 2003b) represents a set of semiconductors companies and aims at identifying and defining a standard interface for the reuse of virtual components or cores. More than two hundred companies are members of the VSI Alliance, from all segments of the electronic industry (ZORIAN; MARINISSEN; DEY, 1998). Seven Working Groups compose the VSI Alliance: Implementation Verification, IP Protection, Manufacturing Test, Mixed Circuits, On-chip bus, SoC Design and IP transfer. The Manufacturing Test and the Verification groups are related to the standardization rules of the virtual components.

The VSIA is meant to promote core-based SoC design by specifying interface standards for design reuse of virtual components, the VSIA term for embedded cores (MARINISSEN et al., 2002). Typically, VSIA endorses existing standards and evaluates emerging ones; if nothing else exists VSIA also develops its own standards or specifications (MARINISSEN et al., 2002). VSIA covers various areas of core-based SoC design. Testing is covered by the Manufacturing-Related Test Development Working Group. This group has in the past specified common test data formats and design-for-testability guidelines for core providers (MARINISSEN et al., 2002).

The VSIA worked in parallel with the IEEE P1500 Working Group to create a gateway for test interoperability meant to be compatible with IEEE 1500. In 2001, it published a specification for a test access infrastructure, which is a prelude to the IEEE 1500 standard and is meant for temporary use until the complete IEEE 1500 standard is eventually finalized and approved (MARINISSEN et al., 2002).

3.4.3 ITC'02 SoC Test Benchmarks

In (MARINISSEN; IYENGAR; CHAKRABARTY, 2002), Marinissen *et al* propose a set of SoC test benchmarks whose main goal is "to stimulate research into new methods and tools for modular testing of SoCs and to enable the objective comparison of such methods and tools with respect to effectiveness and efficiency." The benchmarks format provides the cores test requirements in terms of number and type of tests, number of test patterns, and cores test interface (number and size of internal scan chains, number of test pins, hierarchy level of the core) (MARINISSEN; IYENGAR; CHAKRABARTY, 2003).

Until now, twelve systems compose the ITC'02 SoC Test Benchmarks, among academic and industrial contributions (MARINISSEN; IYENGAR; CHAKRABARTY, 2003):

- u226, from Universidade Federal do Rio Grande do Sul, Porto Alegre, Brazil;
- d281 and d695, from Duke University, Durham, USA;
- h953, from National Tsing Hua University, Hsinchu, Taiwan;

- g1023, from Jiri Gaisler and University of Stuttgart, Stuttgart, Germany;
- f2126, from Faraday Technologies, Hsinchu, Taiwan;
- q12710, from Hewlett-Packard, Shrewbury, USA;
- p22081, p34392, and p93791, from Philips Electronics, Eindhoven The Netherlands;
- t512505, from Texas Instruments, Bangalore, India;
- a586710, from Analog Devices, Austin, USA.

The names assigned to the benchmarks consist of one letter, followed by a number. The letter represents the contributor of the benchmark and it is associated to the system on a first-come-first-serve basis (MARINISSEN; IYENGAR; CHAKRABARTY, 2002). The subsequent number is a positive integer, and indicates the test complexity of the SoC. This number is evaluated using Equation 3.1, as explained in (MARINISSEN; IYENGAR; CHAKRABARTY, 2002). In the equation, consider T the set of module tests. For Test $t \in T$ and corresponding Module $m(t)$, the formula uses the numbers of primary inputs $i_{m(t)}$, primary outputs $o_{m(t)}$, bidirectional terminals $b_{m(t)}$, scan chains $s_{m(t)}$, internal scan chain lengths $I_{m(t),1}, I_{m(t),2}, \ldots, I_{m(t),s_{m(t)}}$, the test pattern count p_t, and the binary parameters su_t and tu_t, indicating, respectively, whether the scan chains are used in t and whether an access mechanism is required for t (t is not a BIST test).

$$\left\lfloor \frac{T \cdot \sum_{t \in T} tu_t \cdot p_t \cdot (i_{m(t)} + o_{m(t)} + b_{m(t)} + su_t \cdot \sum_{x=1}^{s_{m(t)}} I_{m(t),x})}{10,000} \right\rfloor \tag{3.1}$$

The scaling factor 1/10,000 is used to shorten the SoC test complexity number given by Equation 3.1, which is usually large (MARINISSEN; IYENGAR; CHAKRABARTY, 2002).

These benchmarks will be used in this work to validate the proposed test planning techniques. Thus, more detailed information about the benchmarks description and characteristics will be given in Chapter 4, when they are first used.

3.5 Contributions of this Work

The design of a system is a process that involves several constraints, features and trade-offs of different cost factors, such as area, performance, power and test time. Nowadays, test planning has become one of the most expensive steps during the design flow of an electronic circuit built in a core-based design paradigm. Access to embedded cores, the integration of several test methods (not to mention the diversity among the cores themselves) and the optimization of cost factors such as area overhead, test time, and number of test pins in the interface, are just a few of the several problems that need to be tackled during test planning. If extra hardware is ultimately required it enormously complicates the synthesis equations for balancing system constraints.

In general, the system integrator is not a test engineer and it is very difficult to preview the impact of a design decision into the test complexity or, vice-versa, the impact of a test-related decision on the synthesis of the system. Moreover, the system integrator usually looks for the best compromise among the several cost variables involved in the

system synthesis, more than just the full optimization of a single parameter. Although the test costs play an important role in the system cost, test optimization is only one of the variables considered during the system synthesis, which includes area, power, and time-to-market constraints, among others. Therefore, the inclusion of test planning in the very beginning of the system integration is the key for the reduction of the complexity and cost of this task in current and future SoCs.

Until the year 2000, when this thesis was defined, the main drawback of the available solutions was that the test planning task was still conceived late in the design flow, when it is more difficult to change the system structure and constraints. For the test planning approaches that could be performed in the earlier design steps, either only a subset of the cost variables was optimized, or the solution was very dependent on the designer's expertise. Furthermore, each method assumed restricted TAM models on top of which the solution was searched (only test buses, only transparent paths, or only interconnections reuse). Even the methods that contemplate several cost factors at the same time still defined only the TAM based on a previously defined set of test resources, and then optimized the schedule to achieve reduced test times. It was subsequently up to the designer to provide different amounts of test resources (as BIST controllers, TAM bitwidth, number of TAMs or test sets of the cores) and perform a number of searches under a variety of constraints. The test decisions, thus, were mostly based on the designer experience, rather than on the characteristics of the system being designed.

Within this context, this thesis was defined to study a more comprehensive test planning method that could be inserted in the design cycle of the SoC as a support tool for the system integrator. One can observe that this test requirement was also studied by other research groups, and their results (LARSSON; PENG, 2001b; HUANG et al., 2002; KUMAR GOEL; MARINISSEN, 2003) have been published in parallel with the first results of this thesis (COTA et al., 2002, 2003).

This dissertation proposes two test planning approaches to be used by the system integrator, the core provider, and the test engineer as an auxiliary tool in the decision making process of the system development and test. They are meant for the production or off-line testing of SoCs.

The first approach is detailed in Chapter 4, and deals with systems whose cores communicate through traditional connection mechanisms, such as buses and peer-to-peer connections. For those systems, a comprehensive test planning method was developed. This method is based on the definition of a heterogeneous access mechanisms in conjunction with the system test scheduling, and aims at finding the test solution with the best trade-off in terms of test time, pins, and area overheads, considering the system characteristics and power constraints. The main contribution of the proposed methodology is the expansion of the TAM models, its integration with the test scheduling definition, and the consequent possibility of exploration of the system design space during the definition of a test solution.

The second test planning approach is presented in Chapter 5 and deals with systems that use a network-on-chip as the main communication mechanism. For those systems, the network already occupies a considerable area in the chip. Therefore, the inclusion of extra hardware such as test buses for the test of the cores may present additional routing and yield problems. On the other hand, the communication capabilities of the NoC can be used to efficiently transmit test data during the test of the system. The reuse of the NoC resulted in an inexpensive and efficient test solution for NoC-based systems, as the experimental results shown in Chapter 5 will prove.

4 TEST PLANNING AND DESIGN SPACE EXPLORATION IN CORE-BASED SYSTEMS

This chapter presents a comprehensive test planning model for core-based systems, which privileges the reuse of system resources and aims at optimizing a number of cost factors. The main contributions of the proposed model are fourfold:

1. it does not assume a single type of connection for the internal Test Access Mechanisms (TAMs) in the system. Partial test buses are considered, along with functional connections, transparency, and other bypass modes available through the wrapper or the core configuration;

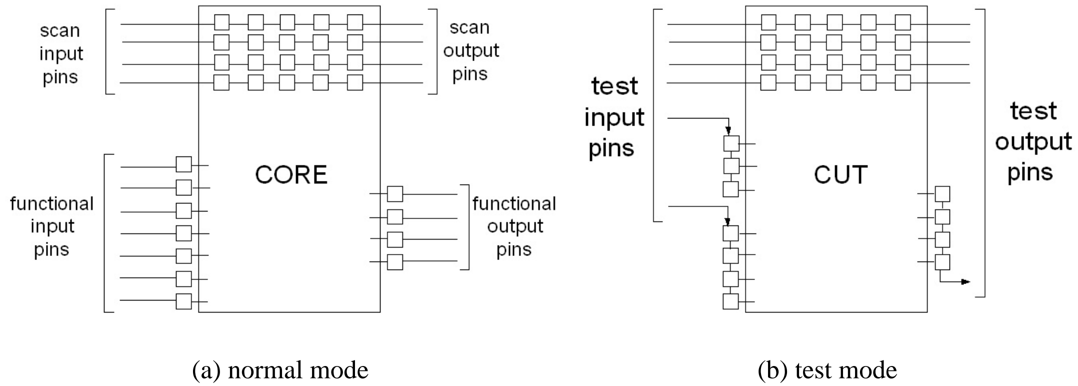
2. the solution does not fully optimize every single core in the system. Instead, the diversity of test requirements among the cores is exploited, by privileging critical cores with more test resources;

3. both the schedule and the global TAM are defined together, and not as independent tasks as in other approaches. This aspect allows the exploration of the design space, so that good compromises among the various trade-offs being sought in the system synthesis can be found.

4. the model represents a step closer to the integration of DFT planning early in the design process, and to a fruitful relationship between core designers and users. The iterative search method provides the system designer with accurate information about critical points in the system being developed. The designer can either suggest specific modifications to the core provider or use this information to guide the synthesis tools in order to find the best solution for the whole system, not only in terms of testing, but with respect to all parameters of the design.

The multi-TAM model and the definition of the scheduling and the global TAM in parallel are the key for the fine-grained exploration of the design space.

Section 4.1 briefly presents the trade-offs involved when using the multiple TAMs model. Then, the problem is formalized and a solution is modeled in Sections 4.2 and 4.3, respectively. The algorithm for test planning is described in Section 4.4. The experimental setup and detailed information about the ITC'02 SoC Test Benchmarks are explained in Section 4.6. At last, the experimental results presented in Section 4.7 show the different trade-offs that can be explored for the same system if a number of cost factors is optimized. Finally, the impact of the system information into the test solution is discussed in Section 4.8.



(a) normal mode

(b) test mode

Figure 4.1: Core interface

4.1 TAM Definition and System Cost Factors

Looking to a core as an isolated entity, the test costs can be easily defined in terms of the number of test patterns, test inputs and outputs, and scan chains. However, when this block is part of a bigger entity, those costs are deeply related to the access mechanism available to this core in the system. Aerts and Marinissen demonstrate this in (AERTS; MARINISSEN, 1998), for the test time cost when a test bus is used as test access mechanism. In this work, we want to expand this reasoning for other types of access mechanisms that can be used within a system.

The TAM model used in this work is defined so that two access paths for each core are defined: a test input and a test output path. Scan and functional pins of a scan-based core are combined into a single set of scan chains in the wrapper, and the functional connections available in the system are reused for the defined set of test pins. This combination consists in the transformation of the functional pins of a core into scan chains, whose length is defined as the maximum length of the internal scan chains of that core. Let a scan-based core *i* with sc_i internal scan chains, I_i inputs, O_i outputs and L_i being the length of the longest internal chain. Two test paths must be defined for this core: one for the inputs, and another one for the outputs. The input bitwidth is defined as $sc_i + \lceil \frac{I_i}{L_i} \rceil$. Analogously, the output bitwidth is defined as $sc_i + \lceil \frac{O_i}{L_i} \rceil$. For example, let us consider the core shown in Figure 4.1(a), with four scan chains of length 5 bits, 7 input, and 4 output pins. The test interface of this core is shown in Figure 4.1(b). The number of input test pins is defined as $4 + \lceil \frac{7}{5} \rceil = 6$ and the number of output pins is $4 + \lceil \frac{4}{5} \rceil = 5$. To simplify, the same number of test pins is considered in both directions when defining a TAM. Thus, for this example, one must find an input and an output access path for 6 test pins.

This pre-processing of scan-based cores is based on the wrapper co-optimization proposed in (IYENGAR; CHAKRABARTY; MARINISSEN, 2001), but implemented in a simpler way. In that work, this transformation takes into consideration the bitwidth of each defined test bus, aiming at finding the best test time for core *i*. Here, this optimization is performed only once, before defining the test path for core *i*. Our basic assumption is that the available functional connections usually outnumber the test pins bitwidth, and the full optimization of the wrapper scan chains is not required. The experimental results show that this is a valid assumption. For cores with external test methods (non scan-based), this pre-processing is not applied, since the reuse of the available functional

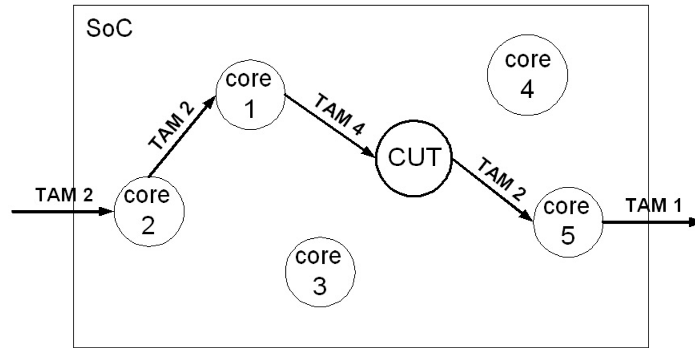


Figure 4.2: Global TAM for a CUT

connections is straightforward. However, all functional pins are propagated as a single set, and not according to the source or target module of each functional port in the interface of the module under test.

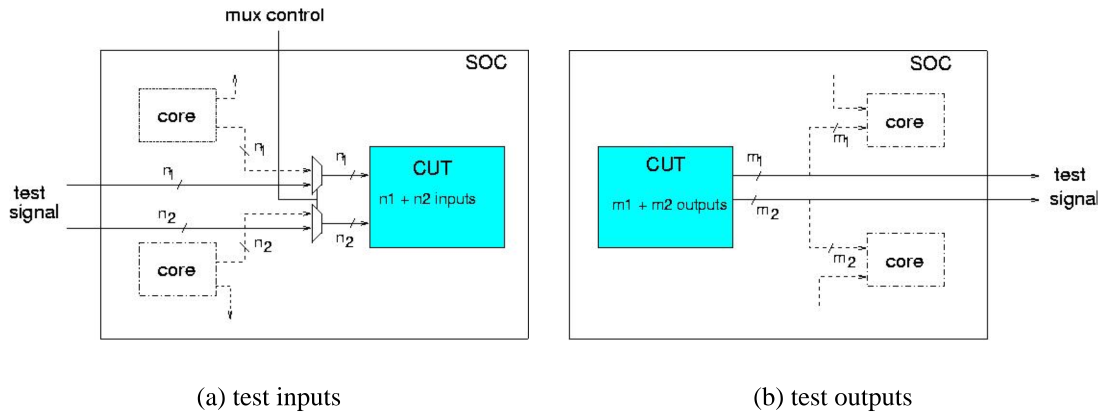
We have defined a multiple yet not exhaustive model for connecting a core under test (CUT) to another core in the system (called here a *neighbor core*), or to the system periphery. This definition is based on the assumption that functional connections already available in the system can be reused during test to reduce test costs. Each type of connection implies distinct costs for the area overhead of the connection, the number of pins for the CUT in the interface, the CUT test time, and the power dissipation of the circuit during test. The area overhead consists of extra wiring (new connections between cores) and extra flip-flops and multiplexers for bypass modes in the wrapper. The number of extra pins for each core is the number of new pins created in the system interface when the TAM for that core is completely defined. The power consumption per cycle is estimated as a function of the extra hardware, in addition to the power dissipated by the core during test.

The complete access mechanism for any core in the system is defined as a series of connections between internal cores, so that two paths, one from the system interface to the core under test (CUT), and another from the CUT to the system boundary are defined. In the next sections, five types of connections are defined. Each pair of cores in the global TAM will be connected by one of these methods, and the impact of each possible local TAM on the core test time and on the area and pin overhead of the system is presented. Notice that a same CUT can be connected to different neighbors by different local TAM models as shown in Figure 4.2.

The the algorithm that selects the type of connection and combines the pairs in a global solution is explained in Section 4.4.

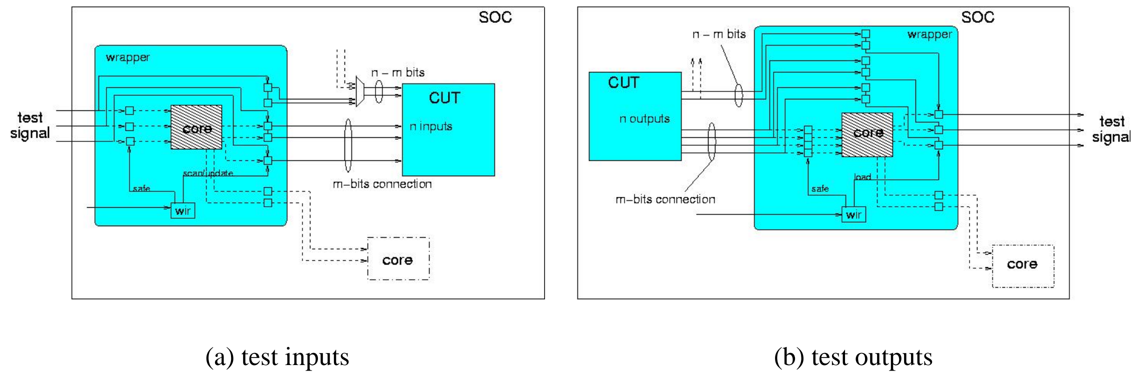
4.1.1 Direct External Access

In this mode, shown in Figure 4.3, no previously existing connection is reused. A direct connection between the CUT interface and the system interface is established. The bitwidth of this connection is assumed to be as large as the number of bits that need to be propagated to the interface. This implies an overhead of n in the number of pins, for n the number of test signals being propagated. The area overhead is proportional to n and to the routing distance from CUT to the system interface. The impact on CUT test time, on the other hand, is the smallest possible, since only one cycle delay is required for test application or observation.



(a) test inputs (b) test outputs

Figure 4.3: Direct external access TAM



(a) test inputs (b) test outputs

Figure 4.4: Access via functional connections

4.1.2 Reuse of Functional Connections

In this TAM, functional connections already present between the CUT and another core are reused, as shown in Figure 4.4. If the bitwidth of the connection is smaller than the number of bits that must be propagated, extra wires are created between the CUT and the neighbor core (Figures 4.4(a) and 4.4(b)). However, this is the only point in the system where a full bitwidth connection is created. A parallel bypass is established between the full bitwidth connection (neighbor-CUT) and one of the functional input (output) ports of the neighbor core. Such a functional port is the one that requires the smallest number of cycles for the serial/parallel conversion operation. The test time of CUT using this connection is $\lceil \frac{n}{m} \rceil L + 1$ cycles per test vector, where m is the number of bits of the selected input of the wrapper of the neighbor core and L denotes the maximum length of the scan chains of CUT. Notice that an additional cycle is considered for the update operation in te scan chains. If the CUT has a non scan-based test, L is 1, representing the wrapper cell associated to the functional pin.

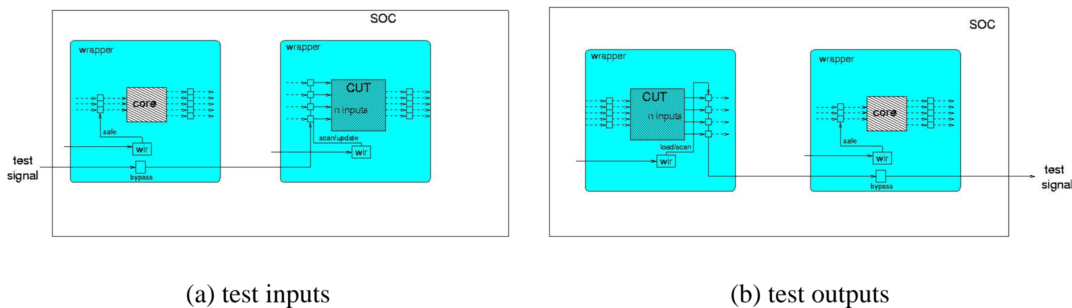
The area overhead of this TAM depends on the width of the existing connection between the CUT and the neighbor core and the distance between them inside the chip. A core whose inputs come mostly from a single source will take advantage of the smaller area overhead for the extra connections required. On the other hand, the bitwidth of the selected neighbor input may be larger than the bitwidth of a pure test bus, reducing the

impact on the CUT test time. Note also that the path from the neighbor core to the system boundaries will be done only for *m* bits, where $m \leq n$. Moreover, the selected port is connected to at least one other core in the system. Hence, the possibility of reuse of existing functional connections from this point on is higher since the number of bits to be propagated (*m* bits) corresponds to the bitwidth of an existent connection. As for the number of pins in the interface, this connection implies no extra pins for the CUT, since it reuses the pins of the neighbor core.

It is important to state that our model assumes internal scan chains of similar length. This may preclude better test time optimizations for scan pins but it simplifies the algorithm by avoiding the problem of finding the best association between available and required pins, as shown in (IYENGAR; CHAKRABARTY; MARINISSEN, 2001). Thus, for example, if 32 test pins are being propagated and the neighbor core has 27 pins that can be used, only 16 out of 27 pins will be considered for the serial/parallel conversion (chains are concatenated in groups of two since $\lceil \frac{36}{27} \rceil = 2$). If the internal scan chains have similar lengths, test time is not deeply affected while total area overhead is reduced.

4.1.3 Use of Serial Bypass

The use of this TAM implies the transformation of all test inputs (outputs) of the CUT in a single external scan-chain. A test vector is loaded serially, and an "update" cycle is required for its application to the core. This TAM is very inexpensive in terms of area, since a single wire traverses between the two cores being connected. On the other hand, it requires the largest number of cycles for test application. This TAM requires $Ln + 1$ cycles to load one test pattern to the CUT. Once this TAM is chosen, only serial bypass is considered from this point on, until the system interface is reached. The pin overhead for this TAM is 1. Figures 4.5(a) and 4.5(b) show the wrapper configuration for test inputs and outputs, respectively.



(a) test inputs

(b) test outputs

Figure 4.5: Access via serial bypass

4.1.4 Use of Transparency Functions

This TAM, shown in Figure 4.6, is considered for test pattern justification in the system when all test inputs of the core under test are connected to the transparent core. The penalty on CUT test time is the number of cycles required for the propagation of the test vectors from the inputs of the neighbor core to its outputs. Transparency is usually the cheapest solution, since no extra hardware is required at the system level. The number of bits to be justified from this core to the system interface is the number of bits that control the transparency function.

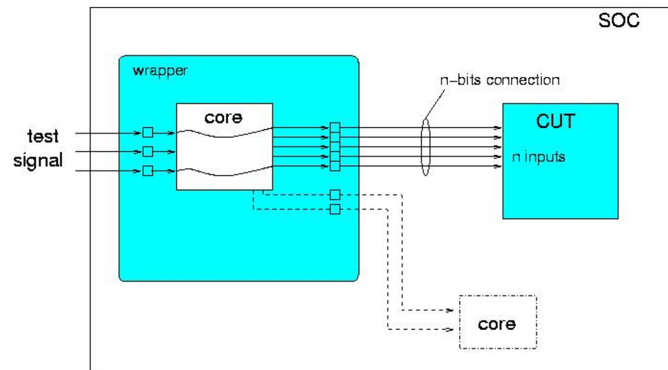
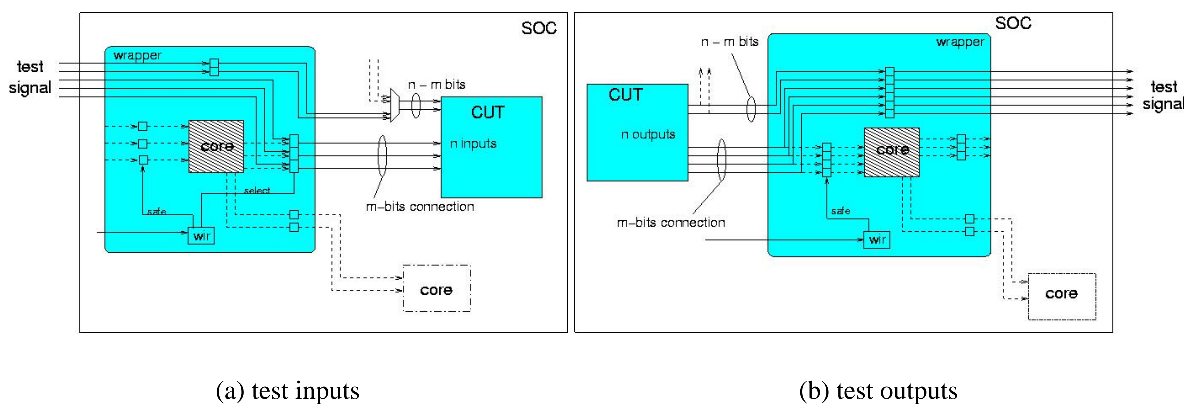


Figure 4.6: Access via transparent core



(a) test inputs (b) test outputs

Figure 4.7: Access via Parallel Bypass

4.1.5 Parallel Bypass

This TAM represents a bus-based access mechanism, where functional connections are reused only in the first level (from the CUT to the first chosen neighbor), as shown in Figure 4.7(a) for the test inputs and in Figure 4.7(b) for the test outputs. From this point on, extra connections of bitwidth n are created accordingly. This is a compromise between the direct access TAM and the TAM presented in Figure 4.4. Although the area overhead for the extra connections may be high for a single core, the possibility of reusing the bus is also high, and the cost is amortized among all cores using the bus. The pin overhead is n for the first core using this TAM, but it is zero for any other core reusing it. In terms of test time, on the other hand, not only does the CUT manifest only a small increase on its test time (1 cycle per neighbor), but also all other cores reusing the bus gain. This type of connection is used when the test time available for the CUT is very restricted.

Using the defined TAM models, the costs of accessing the CUT during test are defined as the path from the CUT to the system boundary is built. The CUT test time is updated for each new connection between intermediate cores in the path, according to the number of extra cycles required at each connection to transmit one test pattern or one test response. The number of extra pins for the CUT is defined by the connection of the last core in the path with the system interface. If this core is already connected to the system interface by functional pins, and those pins are reused, the CUT has no cost in terms of pins. The total area overhead for the test of the CUT is calculated by adding the area of all extra wiring and bypass structures required in the path definition.

4.2 Problem Statement

Considering the different possibilities of test access presented in Section 4.1, the problem being tackled in this work can be stated as follows.

Let the input be:

1. a set $C = \{i, 1 \leq i \leq N\}$ of cores within a system;
2. a set of 6-tuples $I = \{(in, out, s, l, pf, ps)_i, 1 \leq i \leq N\}$, representing the number of external test inputs, external test outputs, scan chains, maximum length of scan chain, number of external testing patterns, and number of scan testing patterns, respectively, for each core in C;
3. a set $T = \{t_i | 1 \leq i \leq 5\}$ of connection models for test pins between two cores as defined in Section 4.1;
4. the functions $CA = Cost_{area}(t_{i,j})$, $CC = Cost_{cycles}(t_{i,j})$, $CPo = Cost_{power}(t_{i,j})$, and $CPi = Cost_{pins}(t_{i,j})$, with $t \in T$ and $i, j \in C$, indicating the cost of a connection between the test pins of core i and the wrapper pins of core j in terms of area, test cycles, power, and pin overhead respectively;
5. a directed weighted graph $G = (V, E)$, with the vertices V representing the cores and the arcs E connecting two vertices if there is a functional connection between them. The weight of each arc is the number of bits connecting the two cores;
6. a triangular matrix containing the distance between any two cores (including the system interface) in the system;
7. a number Po_{max} stating the limit of power consumption during test;
8. a number Pi_{max} stating the limit of extra pins that can be added to the system interface for testing purposes.

We want to find a solution comprised of:

1. A set of k_i-tuples $GT_{in} = \{(t_{a,b}, t_{b,c}, ..., t_{k,interface})_i, 1 \leq i \leq N\}$, for $t_{o,p} \in T$, and $o, p \in C$, representing the access path to the test inputs.The number k_i is the length of the TAM;
2. A set of k_i-tuples $GT_{out} = \{(t_{a,b}, t_{b,c}, ..., t_{k,interface})_i, 1 \leq i \leq N\}$, for $t_{o,p} \in T$, and $o, p \in C$, representing the access path to the test outputs. The number k_i is the length of the TAM;
3. a set $Times = \{(begin, end)_i, 1 \leq i \leq N\}$ representing the starting and ending testing cycles for each core i in the system test schedule.

Thus, the problem is to define a path from the system interface to the input and output of each embedded core and the system test schedule so that the following conditions are respected:

- the total number of extra pins added to the circuit must be less or equal to Pi_{max};
- the power limit Po_{max} must be respected at all times during the system test;

- the total testing time and area overhead are to be optimized in this order.

In the next sections the solution for this problem is formalized and an appropriate heuristic is presented.

4.3 Solution Modeling

The basic assumption used here is that the test access mechanism for a core under test (CUT) is a series of connections between other cores in the system. From this assumption, one can conclude that the global optimization of costs is neither a pure resource allocation nor a pure scheduling problem. Indeed, the total testing time for each CUT can only be defined after the system interface is reached, by adding the extra cycles required at each neighbor core in the path. Also, the complete list of test resources allocated for the CUT is available only after the complete path definition. Thus, one does not know a priori the basic information (task cycles and conflicts over resources) for the application of a traditional scheduling or allocation algorithm if this model is used.

4.3.1 Initial Approach

Since the TAM is defined as a path, the modeling as a shortest path problem is straightforward. If we define the distance in the path as a function of the area, pins, power, and test time overhead for the core under test, traditional shortest path algorithms (SKIENA, 1998) can be used to find the best path from the core periphery to the system interface.

Following this model one can define, for each core in the system, two trees: one for test inputs, and another one for test outputs. Each tree could be the list of all possible paths from the core pins (root) to the system pins (leaves), passing through other cores of the system (internal nodes in the tree), by means of connections modeled in Section 4.1. Each arc connecting two nodes of the tree has a single value representing a function of the costs for area, pins, power and time of that connection; the number of arcs connecting two nodes in the tree is defined by the number of TAM models used. Using this structure, one can find the best solution for each core under test by finding the shortest path from the root to a leaf in each tree.

The modeling suggested in the last paragraph has two problems though:

the size of the tree is impractical: in the worst case, any other core in the system can be used as part of the path for any core under test. This generates, for each core, a tree with the number of nodes defined in Equation 4.1.

$$\textit{total_nodes} = \sum_{1 \leq j \leq N-1} \frac{(N-1)!}{(N-j-1)!} + \sum_{1 \leq j \leq N} \frac{(N-1)!}{(N-j)!} \tag{4.1}$$

As we have two trees per block, the total storage requirement is 2N times the value of Equation 4.1. Even if we can reduce the number of possible paths by defining a sub-set of the cores to be used as part of the path for each CUT, still the size of the tree may be a problem. Moreover, as the complexity of the shortest-path algorithm is at least $O(V^2)$, for V the number of vertices, this solution is clearly impracticable.

the optimization of the system cost is not achieved: the search in each tree aims at optimizing the test costs for each core independently. The global optimization can not be done because each tree is traversed without considering the status of the others. For example, if the best path for core 1 uses cores 2 and 3 before reaching the interface, the cost function of this path is related to core 1 only. This means that the test scheduling for the whole system must be defined after the definition of all test paths, so that cores sharing test resources are not tested at the same time. This kind of solution optimizes the resource allocation, but not the system testing time.

Thus, a modification of this model is required to tackle these two problems. We show in the next subsection how the required search pruning can be done along with the optimization of the system global costs.

4.3.2 Final Approach

Assuming that all test data is available in the core interface using a full bitwidth access mechanism as the one shown in Section 4.1.1, the testing time t_i for each core $1 \leq i \leq N$ is given by Equation 4.2, for L_{max} the length of the longest scan chain, p_i, the number of test patterns, and k_i the number of cycles required for the application of a single test pattern of core i.

$$t_i = \begin{cases} p_i(L_{max}+1) + L_{max} & \text{if core i has scan chains,} \\ p_i.k_i & \text{if core i has no scan chains.} \end{cases} \quad (4.2)$$

This is the minimum testing time for each core. If any other access mechanism is used, as discussed in Section 4.1, core testing time may increase according to the available TAM bitwidth.

Considering the aforementioned Equation 4.2, the best solution for the system in terms of test time is shown in Figure 4.8 for a hypothetical example. In this case, the core with the longest test time ($maxtime = max(t_i), 1 \leq i \leq N$) defines the system test time while the other cores are scheduled in parallel to this one. This solution is impractical in most cases, since the number of extra pins in the system interface and internal wiring overhead is very high as well as power dissipation. On the other hand, one can note in the schedule of Figure 4.8 some unused time slots that could be used to reduce the area and pin costs without affecting the system test time. In other words, let us consider a core j with test time t_j defined for a TAM with cost C_j. The maximum test time for the system is $maxtime$. It is possible to define another TAM of cost $C'_j < C_j$ with corresponding test time t'_j for core j so that $t_j \leq t'_j \leq maxtime$. That is, the new access mechanism for core j has lower cost than the previous one without affecting the total test time for the system. With the different models of TAMs proposed in Section 4.1, any model that increases the test time of a core up to *maxtime* can be used, as long as the TAM cost in terms of area and pins is reduced and the power constraint is ensured.

Applying this reasoning systematically over the optimum solution for testing time, one can explore the design space within this test time while optimizing the test resources allocation.

Two issues arise when applying the above reasoning:

1. the definition of which core will use the idle time slots in the schedule;

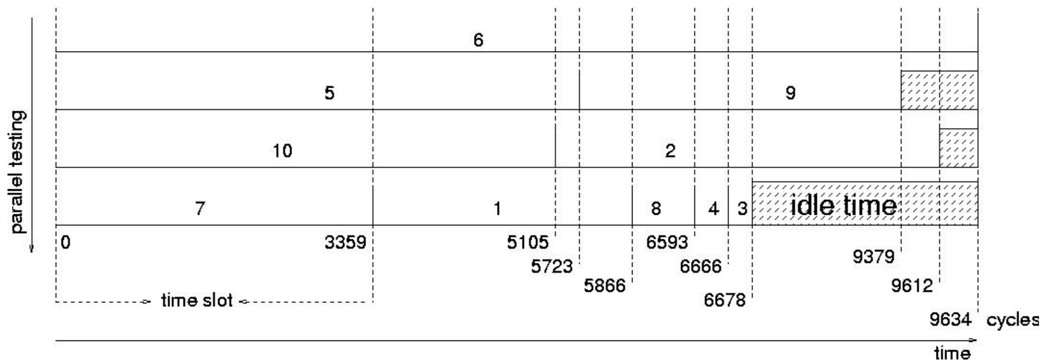


Figure 4.8: Test schedule for minimum test time

2. it is very likely that the initial system time limit, $maxtime$, is too restrictive to accomplish the area, pins and power constraints of the design. Thus, a strategy for the expansion of this limit must also be defined.

The first issue is solved by sorting the cores in decreasing order of test costs, so that cores with a more expensive combination of test requirements (pins, test patterns, scan chain length, etc) have higher priority for test cost reduction. With this criterion, more expensive cores will have more time and resources available.

The second issue is solved if an iterative process of increasing the time limit and evaluating the cost of the resulting schedule can be defined so that the complete conflict list over test resources can be used to guess the new time limit for the system.

Finally, if we concurrently define test scheduling with the access mechanism for each core, the global optimization sought for the system can be implemented. Figure 4.9(a) shows a partial schedule for a hypothetical system, with four cores already scheduled. The conflict list in the right of the Figure shows the conflicts among cores sharing test resources. In this case, core 1 uses core 2 in its path to the system interface while core 2 uses core 1. Core 3 also uses core 2 and it is used by core 5. The unscheduled core 4 has no conflicts yet. Moreover, Figure 4.9(a) shows possible time slots that can be used by core 4 during test.

Figure 4.10(a) presents the first level of one of the two trees representing the possible paths from the CUT 4 to the system interface. In this level, all cores in the system that can be used by core 4 are listed, as well as the system interface itself. The connection to each neighbor can be done in one of the five models defined in Section 4.1. Thus, there are five arcs connecting core 4 to each of its neighbors in the tree. Let us also assume that the costs of those connections are isolated so that any cost can be precisely checked. For each neighbor of core 4, one can check the partial schedule of Figure 4.9(a) and extract the available time slots for the CUT according to the considered neighbor. For example, Figure 4.9(b) presents the result of this verification when core 2 is considered to be used in the path from core 4 to the system interface. Since core 2 is being used by cores 1 and 3, CUT 4 can not be tested together with cores 1, 2 and 3. Thus, the available time slot for the CUT 4, considering core 2 as a test resource, is 100 cycles. The same evaluation is done for each neighbor of core 4, providing the *available time for test* information shown in Figure 4.10(a). Comparing the available time against the CUT test time for each TAM connecting the CUT 4 and its neighbors, one can eliminate the connections that exceed the available time in the current schedule. This is the case, for example, of the connection between core 4 and core 3 in Figure 4.10(b). In this case, all TAMs connecting cores 4 and

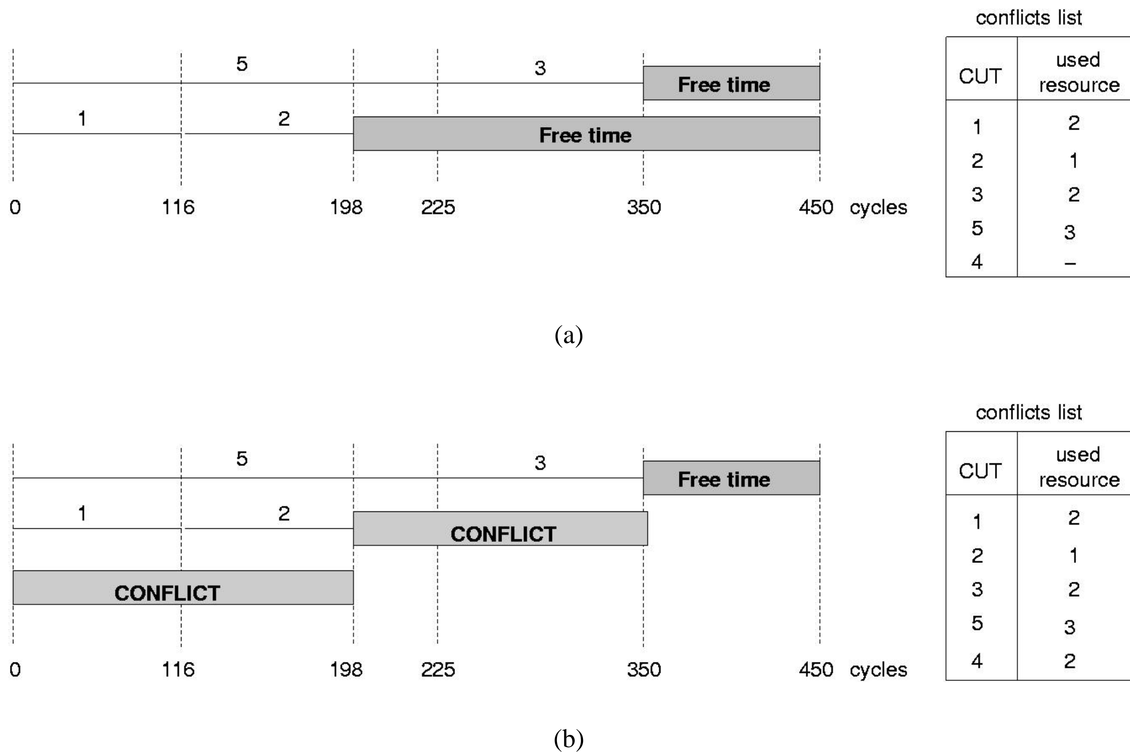


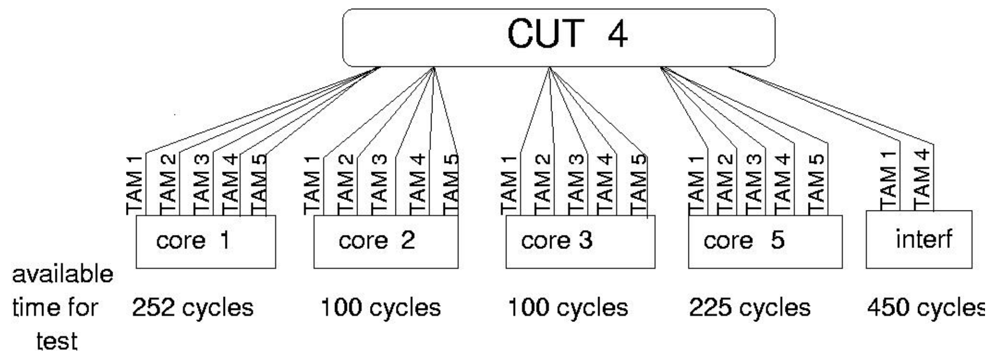
Figure 4.9: Schedule Construction

3 lead to a test time of core 4 larger than the available 100 cycles in the current schedule of Figure 4.9(a). Therefore, core 3 can not be a test resource for core 4. From the remaining possible connections between the CUT and its neighbors, one can choose the cheapest one in terms of area and pins. After this selection, we have the tree configuration shown in Figure 4.10(b). All existing connections in the tree of Figure 4.10(b) represent feasible paths and the cheapest neighbor can be selected for expansion, as shown in Figure 4.10(c). The recursive application of this method until the system interface (a leaf in the tree) is reached gives us the complete test path to CUT 4. After the path for core 4 is completely defined (the two trees associated to the CUT are traversed), the current schedule can be updated.

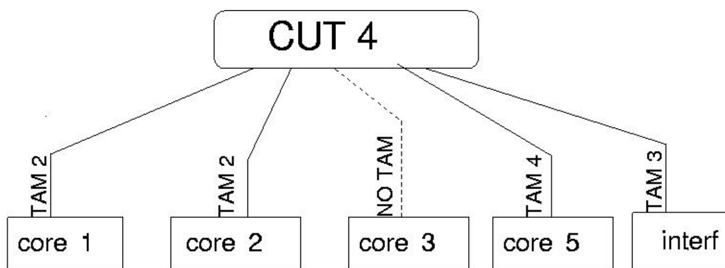
Notice that the cost associated to each connection is still local. The global optimization comes from the partial schedule information. As the path is built in the tree, the list of conflicts over testing resources is updated, as well as the total test cost for the CUT. Moreover, when the schedule is verified, the power consumption of each slot is evaluated so that only slots where the power limit is respected are returned as available.

Summarizing, the following steps are required in the construction of the tree for the shortest path search:

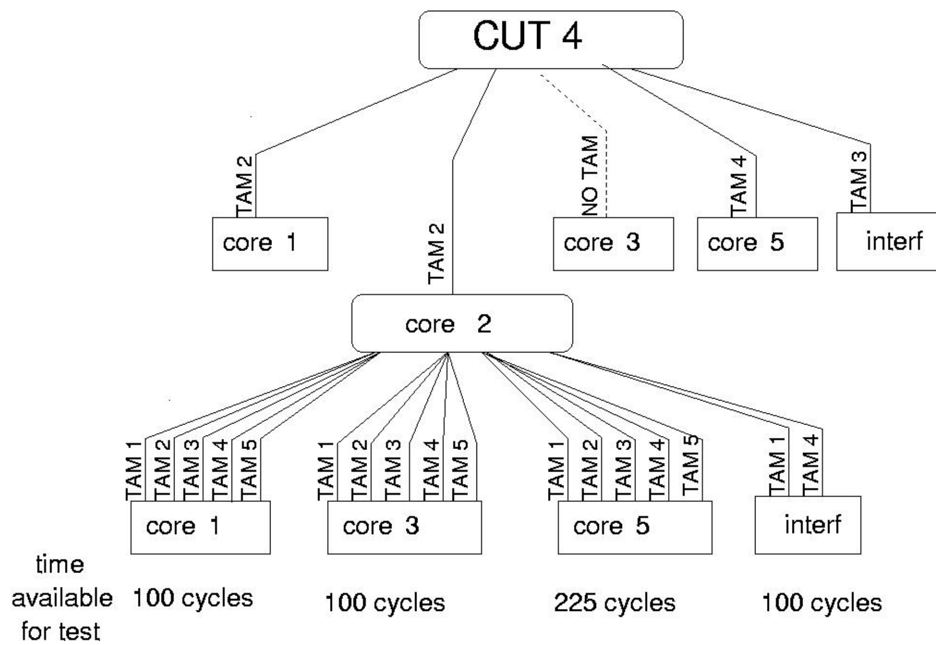
- the cost function of each arc between any two nodes in the tree is defined as a set of four values: time cost, area cost, pins cost and power cost;
- the tree is built on the fly, as the shortest path algorithm is implemented. Only promising nodes are expanded. The definition of a promising node is constructed based on the available time slot extracted from the partial scheduling;
- the shortest-path algorithm is still used to traverse the tree as it is built, but testing



(a) TAMs for CUT neighbors



(b) Best TAM for each neighbor



(c) Best neighbor expansion

Figure 4.10: Tree Construction

time and power are not part of the path cost. Only pins and area are. Instead, time and power overhead are just checked in the partial scheduling so that previously established limits are respected;

- when a core a is used as part of the TAM for a CUT, a conflict is set between these two cores so that they are not tested in parallel. A conflict is also set between any other core using a and the CUT, since they are sharing a test resource.

In the next section, the heuristic used to define the schedule and the resource allocation for the system is presented, based on the modeling just proposed.

4.4 The Proposed Heuristic

For the general case, where several cost factors are being considered at once, the algorithm consists of a search process that starts with the optimum solution for time (minimum test time for the system), with an exclusive TAM provided to the core that sets the time limit for the system. Then, every other core is placed in the schedule in such a way that the best solution for the defined time limit is found. If, after all cores are scheduled, the solution does not satisfy the system constraints, a small perturbation is done in the solution, so that the time limit is slightly increased and a new solution is sought. This process continues until the constraints are attained or until a preset number of iterations is completed.

Figure 4.11 shows the pseudocode for the implemented algorithm. A main loop controls the TAM definition and evaluates the cost of each global solution found, in steps 3 to 19.

The tree for each core is built on the fly (steps 7 to 17 in Figure 4.11), according to the current test scheduling restrictions (step 10) and available TAMs in the system. The cost of each arc in the tree is actually a set of costs, representing the optimization factors being considered. Each core has two sub-trees representing the input and output paths, respectively, and each one is traversed independently. Since the tree represents a core-to-core connection, loops are prevented by the definition of the CUT neighborhood and by the depth of the tree, so that a path to the system interface is always found.

One can observe that the kernel of the algorithm, presented at steps 9 to 14 in Figure 4.11, is the possibility of choosing the best connection (step 13) among the modeled TAMs described in Section 4.1, and the best point to advance the search towards the system interface (step 14). Placement information can accelerate the search by indicating the most promising eligible neighbors for a core according to the location of the pins in the core periphery.

For each eligible neighbor retrieved in step 8 of the algorithm, the schedule is checked to define the amount of time available for the core under test, according to a list of conflicts over test resources. Cores that share a test resource cannot be placed in the same time slot in the schedule. Notice that, if an eligible core is not scheduled yet, a search in the schedule must ensure that a time slot remains for this neighbor to be placed subsequently. Since the cores are scheduled in a decreasing order of cost, less critical cores will have less test resources available, since they have less impact on the solution cost. Thus, they can keep a more expensive TAM (direct access, for example), while other cores are privileged with more time and possibility of reuse of existing resources.

The selection of which core is scheduled first is defined according to the cost of the current solution in terms of pins in the interface, and area and test time for each core.

```
 1. Set initial test time limit for power constraints
 2. While (systems constraints not satisfied) AND (iterations < max_iterations)          C1
 3.   While (there are unscheduled cores)                                                O(N)
 4.       select CUT = critical core for cost factors                                    O(N)
 5.       For each direction of CUT   ports                                              2
 6.           node = CUT
 7.           While (not in the interface) AND ( accumulated cost <= current cost)       C2
 8.               neighbors = retrieve (CUT neighborhood) AND (node neighborhood)        O(1)
 9.               For (each core in neighbors)                                           O(N)
10.                   time_slot = available time slot in the current schedule            C2(O(2N))
11.                   For (each modelled TAM)                                            T
12.                       evaluate costs of TAM connection CUT−neighbor                  O(1)
13.                   Select best TAM CUT−neighbor                                       O(1)
14.               Select best neighbor for CUT                                           O(N)
15.               accumulated cost += cost CUT−selected neighbor                         O(1)
16.               node = selected neighbor
17.               GO TO step 7.
18.       Insert CUT in the schedule                                                     C2(O(2N))
19.       Update conflict list                                                           O(N)
20.   IF (constraints not satisfied)
21.       set test time limit = infinity
22.       select CUT = critical core for cost factors                                    O(N)
23.       Find a less expensive TAM to CUT                                               O(N 3)
24.       Schedule  CUT
25.       Set new test time limit for the system                                         O(1)
26.       GO TO Step 5.
```

Figure 4.11: Pseudo-code of proposed model

$$
\begin{aligned}
H(N) &= C_1\left[O(N)\left[2C_2\left[O(N)\left[C_2O(2N)+TO(1)\right]+O(N)\right]+C_2O(2N)+O(N)\right]\right] \\
&= 4C_1C_2^2O(N^3)+2C_1C_2(T+2)O(N^2)+6(C_1C_2+1)O(N) \\
&\quad K_1 = 4C_1C_2^2 \\
&\quad K_2 = 2C_1C_2(T+2) \\
&\quad K_3 = C_1(6C_2+1) \\
H(N) &= K_1O(N^3)+K_2O(N^2)+K_3O(N) \\
H(N) &= O(N^3)
\end{aligned}
\tag{4.3}
$$

For different combinations of optimization factors, different costs are considered in selecting a core to be placed in the schedule. The most critical core (the most expensive one according to the cost factors being optimized) is selected from the list of unscheduled cores. Similarly, a TAM is defined first for the direction (input/output) that is using more resources in the current solution.

The small perturbation in a current solution that allows a refined search for the global minimum, is shown at steps 20 to 26 of Figure 4.11. It consists of selecting the most critical core, that is, the core that requires more test resources in the current solution. For this core, a new solution is searched as described at steps 7 to 17. However, at this point no scheduling checking is done (step 10). Any possible solution that reduces the cost of the available TAM for the critical core is considered. For test time minimization, the selection of the best neighbor at any level of the tree (step 14) is performed so that the neighbor that implies the smallest increment on the core test time is selected. Then, based on the conflicts of the current global solution, a new test time limit is calculated (step 25) such that current optimizations can be retained if necessary. The schedule is re-initialized and the search starts from the beginning (step 3).

4.5 Heuristic Complexity

As it can be seen in Figure 4.11 five loops (lines 2, 3, 5, 7, 9, and 11) define the overall complexity of the proposed heuristic. The complexity associated with each block of the algorithm can be seen at the end of each line in Figure 4.11, for N the number of cores in the system. The complexity of loops at lines 3 and 5 is straightforward. The block within the loop of line 7 is executed while the system interface is not reached, that is, it is dependent on the depth of the tree. The maximum depth of the tree is a parameter of the algorithm, defined by the designer according to system characteristics. It can be defined, for example, based on the maximum number of cores serially connected in the application or based on the placement, to avoid the definition of very long paths. Thus, this block is executed at most C_2 times. In line 9, any core in the system can be used in the TAM for any CUT, in the worst case. Therefore, this loop is $O(N)$. Line 10 represents the search in the partial schedule for an available time slot. This function must ensure that all unscheduled cores in conflict with the CUT can be placed in the current *maxtime* limit. The CUT will be using at most C_2 other cores to form its TAM. Each core inserted in the schedule adds one or two time slots, depending on the insertion point. In the worst case, the final schedule has $2N$ slots to be searched. Thus, the complexity of the search

in the schedule is $C_2O(2N)$. Line 11 is executed exactly T times, for T the number of TAM models being used (maximum five in our case). The selection of the best node to be expanded in the tree is $O(N)$, if all $(N-1)$ cores can be used as neighbors for CUT, although this number diminishes as the tree is built. The insertion of the CUT in the current schedule has the same complexity of the search for a time slot in the scheduling performed in line 10, that is, $C_2O(2N)$. Finally, the algorithm is executed at most C_1 times (step 2), if no solution satisfying the system constraints is found. Constants C_1 and C_2 are defined by the system integrator.

Then, combining the complexity of lines 3 to 19 of the algorithm as shown in Equation 4.3, one can conclude that the overall complexity of the heuristic is $O(N^3)$.

4.6 Experimental Setup

A prototype tool based on the described heuristic was implemented using Matlab (MATHWORKS, 1997), and it is called **Re**use-**Ba**sed **Te**st Planning (ReBaTe). In Section 4.7, the results provided by this tool for the ITC'02 SoC Test Benchmarks (MARINISSEN; IYENGAR; CHAKRABARTY, 2002, 2003) are presented. The benchmark set is detailed in the next section, as well as the methods used to estimate the system information that are not available in the benchmarks description.

In the CD-ROM that accompanies this manuscript, one can find the current version of the ReBaTe tool, as well as the description and complete set of solutions for the available ITC'02 benchmarks.

4.6.1 ITC'02 SoC Test Benchmarks

Each benchmark in the ITC'02 set is described as follows (MARINISSEN; IYENGAR; CHAKRABARTY, 2002):

1. The SOC name according to the ITC'02 benchmarks naming convention;
2. the total number of modules in the SOC;
3. global settings that specify whether or not the optional data for layout position and power dissipation are provided;
4. for each module in the SOC (all cores are modules, and also the SOC is considered to be a module):
   (a) the level in the design hierarchy;
   (b) the number of input, output, and bidirectional terminals;
   (c) the number of scan chains and their lengths;
   (d) the absolute layout location of the core (optional);
   (e) the number of tests, and per test:
      i. whether or not this test uses the module-internal scan chains and/or the core-external Test Access Mechanism (TAM).
      ii. the number of test patterns
      iii. the power dissipation (optional).

Table 4.1: Some Characteristics of the ITC'02 Benchmarks (MARINISSEN; IYENGAR; CHAKRABARTY, 2002)

SoC	Number of				
	Modules	Levels	$\sum$I/Os	$\sum$SFFs	$\sum$Test Patterns
u226	10	2	376	1,040	5,148,569
d281	9	2	2,931	882	8,818
d695	11	2	1,845	6,384	881
h953	9	2	929	4,657	1,100
g1023	15	2	3,707	1,546	2,349
f2126	5	2	1,597	13,996	962
q12710	6	2	13,167	12,991	4,612
p22810	29	3	4,283	24,723	24,890
p34392	20	3	2,057	20,948	66,349
p93791	33	3	6,943	89,973	22,987
t512505	31	2	8,663	68,051	10,479
a586710	8	3	3,755	37,656	10,850,894

Table 4.1 shows the main characteristics of the available SoC benchmarks.

Notice that in order to model such systems in the test planning tool, one needs some important information regarding the system itself that is not available in the current benchmark format. Although the information per core is precise (number of vectors, inputs and outputs, scan chains, etc) the system information is optional. As a matter of fact, none of the current benchmarks provides the placement information, for example, and only one system (h953) has the power consumption data for each embedded module. However, this system-related information is important in the proposed method to evaluate the area overhead and to allow the reuse of available resources. Therefore, for each system, hypothetic data was generated based on the information currently available.

Firstly, the area and dimensions of each core were estimated as a function of the number of inputs, outputs, and scan flip-flops, as shown in Equation 4.4. Terms α and β are used to increase or reduce the size of the core if necessary. Variables nb_{ff} is the number of scan flip-flops informed in the benchmark description. The number of gates of the core is estimated as a function of the number of inputs and outputs of the module (variables $inputs_{nb}$ and $outputs_{nb}$ in Equation 4.4). The areas of one gate ($gate_{area}$) and one flip-flop (ff_{area}) are defined as technology-dependent constants, and are also used to evaluate the area overhead of the TAMs defined during the test planning.

$$core_{area} = \alpha(nb_{ff} \cdot ff_{area}) + \beta(inputs_{nb} + outputs_{nb}) \cdot gate_{area} \quad (4.4)$$

The power consumption of the cores during test is another information of the available benchmarks that must be estimated. Equation 4.5 models the dynamic consumption per cycle of a logic block. C_L is the load capacitance (technology-dependent constant), T is the test clock period. Variable nb_{gt} is the estimated number of gates of the core, calculated as part of Equation 4.4. sw_{ff} and sw_{gt} are the switching factors for each type of gate, respectively. These factors depend on the test vectors defined for each core, which is not available for the ITC'02 benchmarks. Thus, a factor of 0.5 is used in the experiments. Notice that for the flip-flops, there is a constant switching factor caused by the clock, in addition to the eventual switching in the stored bit value.

$$core_{power} = C_L * Vdd^2 * \frac{1}{T} * [(sw_{ff} + 1) * nb_{ff} + sw_{gt} * nb_{gt}] \quad (4.5)$$

The power consumption of the wrapper of each core is also evaluated using Equation 4.5 by replacing the number of flip-flops and gates accordingly, and it is added to the power consumption of the core. The consumption of the core and of the wrapper given by Equation 4.5 is considered the peak consumption of the module to process all test patterns.

The functional connections among the embedded cores are randomly assigned and, based on these assignments, a floorplanning is devised. For some systems, different configurations were defined: some considering pure peer-to-peer connections and others considering functional buses connecting bidirectional pins.

For the placement information, a simulated annealing placement tool is used to define a hypothetic placement for the cores inside the chip. This placement tool aims at minimizing the total chip area and the routing area among cores. Thus, the functional connections previously defined are used as input parameters to the placement algorithm. Clearly, this might result in non optimized circuits, but the information obtained is still valid for the validation of the tool.

As one will note next, the proposed tool is very sensitive to those estimated system characteristics, and the results must be evaluated accordingly. For system u226, for example, the actual functional connections are available, and a comparison between the result for the estimated connections and for the real ones shows that the result for the real systems is probably better than the result presented here. In Section 4.8 the impact of the system information on the test solution is discussed in more detail.

Furthermore, it is important to clarify that the area overhead calculated by the ReBaTe tool is a function of the area of a transistor, which is ultimately defined for a specific technology. In the experiments presented in this dissertation, the typical values associated to a 0.8μ technology are used. Moreover, the area overhead is evaluated according to the area defined for the system, which includes the estimated area of each core and of the functional connections. Therefore, for the real systems, the area overheads can be different from the ones presented here.

4.7 Experimental Results

In the sequel, some results for the currently available ITC'02 SoC Test Benchmarks (MARINISSEN; IYENGAR; CHAKRABARTY, 2003) are presented, showing the variety of system configurations and test solutions that can be explored using the proposed test planning tool.

4.7.1 Benchmarks d695, g1023, f2126, q12710, and t512505

This section groups the benchmarks d695, g1023, f2126, q12710, and t512505 of the ITC'02 suite. These systems are the simplest to be modeled in the proposed test planning tool: there is only one level of hierarchy, each core has only one test, all tests require an access mechanism (no BIST testing), and there is no test defined at system level (no specific UDL testing). Hence, each module described in the system can be directly modeled as one CUT in the test planning tool, with its list of test requirements (test pins, number of vectors, etc) and characteristics (power consumption during test, connections to other modules, location in the chip, and so on).

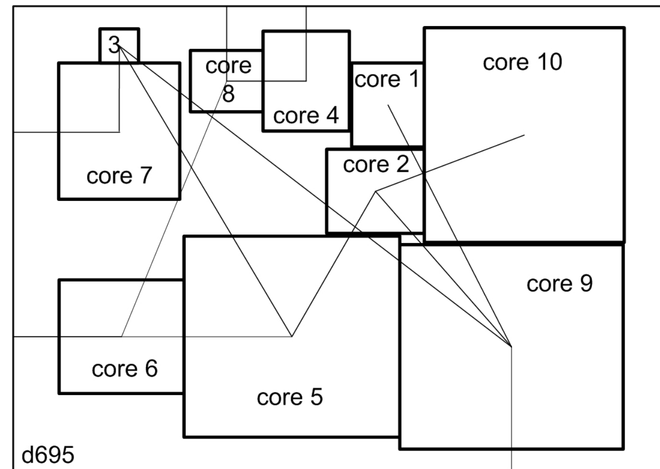


Figure 4.12: Floorplanning of benchmark d695

Benchmark d695 is composed by a set of ISCAS'85 and ISCAS'89 circuits as cores. It was originally proposed in (IYENGAR, 2001) and later modified to be inserted into the ITC'02 set. The connections and placement of this system are shown in Figure 4.12.

Table 4.2 presents a summary of the test requirements of the embedded cores in d695. The first column indicates the embedded cores, as described in the benchmark. Module 0 (zero) refers to the system level or UDL. The number of scan chains originally defined for the module is shown in the second column, and the maximum length of the original scan chains is shown in the third column. The number of functional inputs and outputs is presented in the fourth column. The fifth column shows the number of test pins for which an access mechanism is defined, as explained in Section 4.1. For example, for module six, the functional inputs were concatenated into two scan chains of length 31 ($\lceil \frac{62}{41} \rceil = 2$ and $\lceil \frac{62}{2} \rceil = 31$), resulting in eighteen input test pins (16 pins for the original scan chains and 2 pins for the new chains formed with the functional pins). Similarly, the functional outputs were transformed into four chains of length 38, resulting in 20 output test pins. To simplify further serializations during the TAM definition, the larger number of test pins (20, in this case) is considered in both directions. Finally, the last column indicates the number of test patterns of each module.

The description of this benchmark does not specify the number of inputs and outputs of the system. The values presented in Table 4.2 were randomly defined, together with the connections among the embedded cores.

Table 4.3 presents some resulting test costs for the benchmark d695 presented in Figure 4.12. The second column shows the list of costs optimized during the search for the test solution: PT stands for pin count and test time optimization, PTA represents pin count, test time and area overhead optimization, and PTAP represents pin count, test time, and area overhead optimization, along with power dissipation controlling, with the power limit indicated in parenthesis. The limit for the number of extra pins at the system interface is set to zero. If a solution for this number of pins is not found, the best solution among all partial solutions found by the tool is shown.

The power consumption limit is defined as a function of the power consumption of the system cores during test. This function is defined as a percentage of the sum of the power consumption of all cores. Thus, for example, a power limit of 50% indicates that the power consumption limit corresponds to half of the sum of the power consumption of all cores in test mode. Notice that in a real case, the designer can define any power limit.

Table 4.2: Test requirements of benchmark d695

Module	Nb. of Scan chains	Scan chain depth	Nb. of I/Os	I/O test pins	Nb. of test patterns
0	0	0	100/141	0/0	0
1	0	0	32/32	32/32	12
2	0	0	207/108	207/108	73
3	1	32	34/1	2/2	75
4	4	54	36/39	5/5	105
5	32	45	38/304	39/39	110
6	16	41	62/152	20/20	234
7	16	34	77/150	21/21	95
8	4	46	35/49	6/6	97
9	32	54	35/320	38/38	12
10	32	55	28/106	34/34	68

Table 4.3: Test results for benchmark d695

Experiment	Optimized factors	No. of extra pins	Test time	Area overhead
1	PT	138	9,869	12%
2	PT	0	21,986	18.2%
3	PT	169	11,453	9%
4	PTA	182	9,869	7%
5	PTA	3	37,089	3.8%
6	PTA	38	24,323	2%
7	PTA	8	16,566	5.0%
8	PTAP (80%)	3	37,089	3.8%
9	PTAP (50%)	260	9,869	8%
10	PTAP (50%)	1	31,021	7.6%
11	PTAP (50%)	56	26,460	2%
12	PTAP (50%)	7	24,323	4.0%
13	PTAP (30%)	367	14,219	5%
14	PTAP (30%)	94	25,147	5.3%
15	PTAP (30%)	136	26,143	2%

Optimization factors: PT = pins&time,
PTA = pins&time&area, PTAP = pins&time&area&power

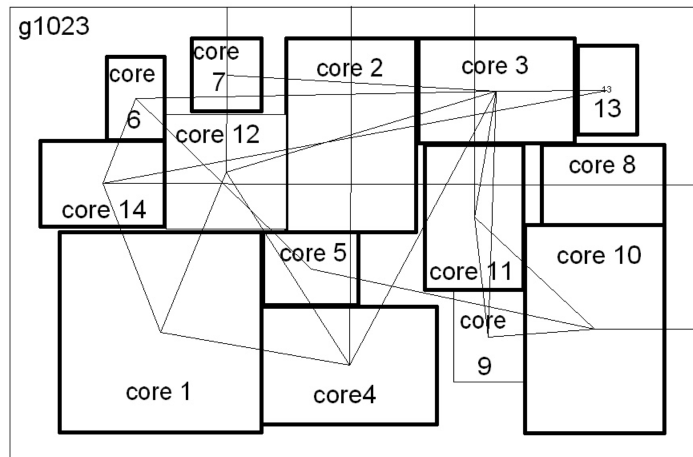


Figure 4.13: Floorplanning of benchmark g1023

Experiments 1 to 15 in Table 4.3 show the different trade-offs that can be considered for the system d695 according to the cost factors being optimized and the system constraints. For each set of optimization factors, the solutions with smallest test time, pin count, and area overhead are presented, respectively. For some cases, another intermediary solution is also presented.

The area overhead can be quite high if it is not considered during the test planning, as shown by experiments 1 to 3. If area is also an issue, the system integrator has other choices, exemplified in experiments 4 to 7. He/she can choose keeping the reduced test time while increasing the system pin count and the area overhead (experiment 4), keeping the reduced pin count and increasing the test time (experiment 5), or keeping the reduced area overhead but increasing the test time (experiment 6). A compromise between the three costs is presented in experiment 7.

Experiment 8 shows that the a power limit of 80% does not compromise the test solutions previously found. Reduced limits, on the other hand, imply extra costs in one of the test parameters. Experiments 9 to 12 represent some of the possible solutions for this system when a power consumption limit of 50% of the sum of the power consumption of all cores is considered. Experiment 12 is a solution that presents a trade-off among the test costs while still respecting the system power constraints. One can notice that the tool could find a solution for a power limit of 50% that is quite close to the solution where no power constraints are considered. However, Experiments 13 to 15 show that more restricted power limits have a great impact on the test solution. To reduce the power consumption, less wrappers are used to form a TAM, and more pins need to be created in the system interface, which reduces the system test time and keeps the area overhead at lower levels.

Benchmark g1023 is composed of fourteen modules, and the number of pins at system-level is defined in the benchmark description. The floorplanning and connections defined for this system are shown in Figure 4.13 and the test requirements of the modules that compose this system are summarized in Table 4.4. Notice the third value in the number of interfaces of module 0 (system interface). This value indicates the number of bidirectional pins present in the system interface.

One can observe in Table 4.5 that the optimization of area in this system implies an increase in the number of pins with a subsequent reduction in the test time. Experiment 7 shows, on the other hand, that better trade-offs for the system can be found if more

Table 4.4: Test requirements of benchmark g1023

Module	Nb. of Scan chains	Scan chain depth	Nb. of I/Os	I/O test pins	Nb. of test patterns
0	0	0	4/63/53	0/0	0
1	14	43	139/273	21/21	134
2	2	84	221/215	5/5	74
3	1	53	192/171	5/5	57
4	4	54	145/155	7/7	268
5	4	32	32/27	5/5	51
6	2	47	20/18	3/3	36
7	2	47	20/18	3/3	34
8	2	52	63/80	4/4	31
9	1	64	56/34	2/2	68
10	1	13	301/377	30/30	29
11	1	9	145/191	23/23	15
12	1	13	157/161	14/14	16
13	0	0	58/64	58/64	512
14	0	0	140/114	140/114	1024

Table 4.5: Test results for benchmark g1023

Experiment	Optimized factors	No. of extra pins	Test time	Area overhead
1	PT	197	14,794	25%
2	PT	0	86,973	18%
3	PT	25	143,796	15%
4	PTA	241	14,794	3%
5	PTA	3	157,120	3.2%
6	PTA	5	172,588	1%
7	PTA	21	38,773	2%
8	PTAP (80%)	3	157,120	3.2%
9	PTAP (50%)	241	14,794	3%
10	PTAP (50%)	2	51,383	2.6%
11	PTAP (50%)	30	149,895	1%
12	PTAP (50%)	4	39,844	3%
13	PTAP (30%)	279	20,642	3%
14	PTAP (30%)	17	48,501	2.3%
15	PTAP (30%)	108	55,803	1%

Optimization factors: PT = pins&time,
PTA = pins&time&area, PTAP = pins&time&area&power

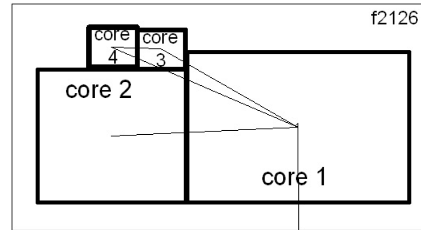


Figure 4.14: Floorplanning of benchmark f2126

Table 4.6: Test requirements of benchmark f2126

Module	Nb. of Scan chains	Scan chain depth	Nb. of I/Os	I/O test pins	Nb. of test patterns
0	0	0	52/140/196	0/0	0
1	8	1,000	356/529	9/9	334
2	16	319	85/139	17/17	422
3	1	452	30/20	2/2	103
4	1	452	30/20	2/2	103

factors are optimized at the same time. The reason for this is that a new optimization factor changes the space where the search takes place, avoiding the local minimum found in the previous situation. Indeed, although the number of pins and test cycles is very similar for experiments 5 and 10, the new pins are associated to distinct cores in each solution, and the test schedule is such that the power limit is respected.

System f2126 has only four modules, as shown in Figure 4.14. However, the complexity of the test of this system lies on the size of the embedded cores, as the test requirements in Table 4.6 show.

Table 4.7: Test results for benchmark f2126

Experiment	Optimized factors	No. of extra pins	Test time	Area overhead
1	PT	34	335,334	6%
2	PT	18	430,642	2.8%
3	PT	19	518,858	2%
4	PTA	37	335,334	4%
5	PTA	17	566,105	2.4%
6	PTA	20	471,363	2%
7	PTAP (80%)	18	518,858	2.5%
8	PTAP (60%)	56	470,693	6%
9	PTAP (60%)	19	518,858	2.3%
10	PTAP (60%)	36	470,693	5%

Optimization factors: PT = pins&time,
PTA = pins&time&area, PTAP = pins&time&area&power

The test results presented in Table 4.7 for this system show that, despite the reduced number of resources, there is a considerable variability of the solutions sought by the test planning tool. The area overhead does not change significantly from one solution to

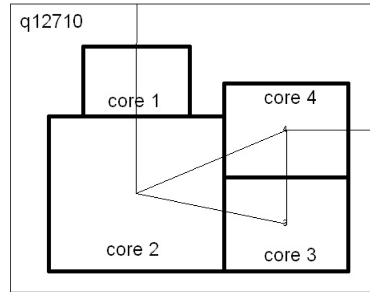


Figure 4.15: Floorplanning of benchmark q12710

Table 4.8: Test requirements of benchmark q12710

Module	Nb. of Scan chains	Scan chain depth	Nb. of I/Os	I/O test pins	Nb. of test patterns
0	0	0	1594/1697	0/0	0
1	3	971	655/777	4/4	852
2	4	1,689	3784/3379	7/7	1,314
3	3	1,297	970/1316	5/5	1,223
4	3	1,297	970/1316	5/5	1,223

another, but the number of pins and the test time do. This is caused by the variability of the test requirements among the embedded cores. Furthermore, the minimum power limit for this system is 60%, since the power consumption of Module 1 during test has this value.

System q12710 is another example of a small but complex SoC. It is also composed of only four, but large, modules. Figure 4.15 shows the floorplanning for this benchmark while Table 4.8 present the system test requirements. Again, as the number of interface pins of the system is not provided in the benchmark description, the values shown in Table 4.8 are synthetic and were defined based on the number of pins in the embedded cores.

For benchmark q12710 the variability of the solutions is very reduced, as shown in Table 4.9. The area overhead is very close to zero in all cases, and the solution found when pins, time, and area are optimized is equal to the solution found for pins and time optimization only. Although the power limit of 80% increases the system test time by 34% (Experiment 7) if compared to Experiment 5, better solutions are found for more reduced power limits (experiment 8). One can observe that the test requirements of the cores of this benchmark are very similar, which contributes, for the reduction in the search space.

Similarly to system f2126, the minimum power limit for benchmark q12710 corresponds to the power consumption of Module 2, which evaluates to 53% of the sum of the consumption of all cores.

Benchmark t512505 is composed of 31 cores, as shown in Figure 4.16, with different test complexities, as shown in Table 4.10. The test results for this system are presented in Table 4.11.

For this benchmark, the variability of the test solutions is high, but none of them presents a pin overhead close to zero. Indeed, the best solution for the number of pins (Experiment 13) presents a very high test time. On the other hand, the best solution

Table 4.9: Test results for benchmark q12710

Experiment	Optimized factors	No. of extra pins	Test time	Area overhead
1	PT	23	2,222,349	0%
2	PT	0	4,645,404	0%
3	PT	4	3,817,402	0%
4	PTA	23	2,222,349	0%
5	PTA	0	4,645,404	0%
6	PTA	4	3,813,548	0%
7	PTAP (80%)	0	6,236,421	0%
8	PTAP (53%)	1	3,313,904	0%

Optimization factors: PT = pins&time,
PTA = pins&time&area, PTAP = pins&time&area&power

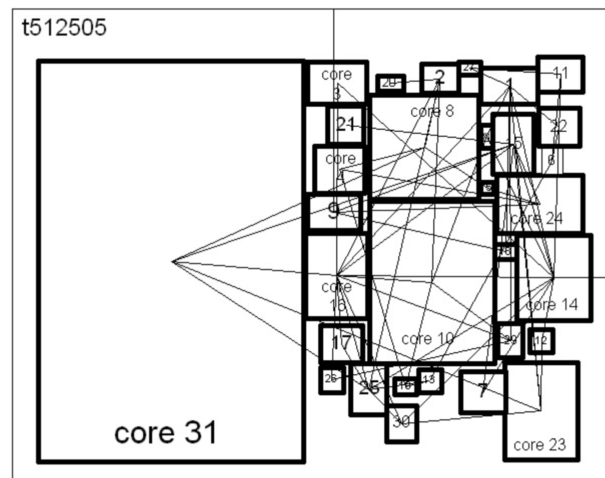


Figure 4.16: Floorplanning of benchmark t512505

for time implies a considerably large number of test pins and area overhead. The best compromise found for this system is, therefore, the one presented in Experiment 8, where the number of extra pins is very close to the minimum found by the tool, the test time is only 16% higher than the minimum, and the area overhead is only 1%. Thus, if the power limit of 80% is acceptable to this application, this solution can be used.

4.7.2 Benchmark h953

System h953 is very similar to the benchmarks presented in Section 4.7.1 in terms of test complexity. It contains only one level of hierarchy, each core has only one test, all tests require an access mechanism (no BIST testing), and there is no test defined at system level. The new aspect of system h953 is that the power consumption of the embedded modules is available in the system description. This system is the only benchmark that brings this information. Therefore, the power consumption of the cores does not need to be estimated as explained in Section 4.6.1, as it happens for the other benchmarks. However, the power consumption is given as a pure number, with no other information, such as whether the consumption is measured per cycle or if it is a peak or average consumption. Hence, even for this benchmark some assumptions listed in Section 4.6.1 about the power

Table 4.10: Test requirements of benchmark t512505

Module	**Nb. of Scan chains**	**Scan chain depth**	**Nb. of I/Os**	**I/O test pins**	**Nb. of test patterns**
0	0	0	15/13/132	0/0	0
1	1	389	206/151	2/2	157
2	1	104	199/89	3/3	330
3	1	904	61/38	2/2	154
4	1	740	114/92	2/2	408
5	1	10	9/6	2/2	3
6	1	154	28/22	2/2	127
7	1	514	122/36	2/2	608
8	3	1,473	106/147	4/4	1,025
9	1	530	82/122	2/2	195
10	8	1,264	64/113	8/8	788
11	1	530	75/34	2/2	188
12	1	53	46/74	3/3	42
13	1	94	56/37	2/2	68
14	1	1,225	751/381	2/2	278
15	1	386	406/132	3/3	151
16	1	154	850/897	7/7	370
17	1	131	303/134	4/4	80
18	1	73	30/20	2/2	153
19	1	68	29/21	2/2	79
20	1	68	29/21	2/2	77
21	1	540	23/19	2/2	242
22	1	540	23/19	2/2	233
23	2	1,372	99/124	3/3	532
24	1	1,669	182/129	2/2	429
25	1	190	352/136	3/3	148
26	0	0	89/127	89/127	13
27	0	0	46/53	46/53	10
28	0	0	5/2	5/2	3
29	0	0	150/157	150/157	67
30	1	302	51/57	2/2	151
31	28	1,550	211/316	29/29	3,370

Table 4.11: Test results for benchmark t512505

Experiment	Optimized factors	No. of extra pins	Test time	Area overhead
1	PT	71	5,228,420	7%
2	PT	28	5,751,444	8%
3	PT	30	5,745,321	5%
4	PTA	92	5,228,420	0%
5	PTA	13	7,485,064	1%
6	PTA	72	6,605,632	0%
7	PTA	17	6,490,281	1%
8	PTAP (80%)	14	6,092,953	1%
9	PTAP (50%)	91	5,569,563	0%
10	PTAP (50%)	19	6,267,593	1.6%
11	PTAP (50%)	26	5,959,937	0%
12	PTAP (30%)	106	6,740,743	0%
13	PTAP (30%)	8	14,278,685	1.6%
14	PTAP (30%)	31	8,213,834	2%

Optimization factors: PT = pins&time,
PTA = pins&time&area, PTAP = pins&time&area&power

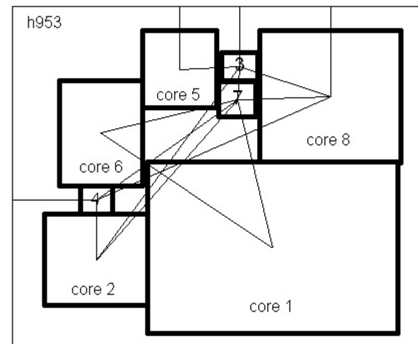


Figure 4.17: Floorplanning of benchmark h953

profile of the cores were required.

Figure 4.17 shows the floorplanning of this system and Table 4.12 presents the test requirements of the embedded modules. Table 4.13 shows some test solutions found for this benchmark.

One can observe for system h953 the reduced impact of the power constraints in the test solution, as very similar solutions are found for power limits of 80% and 50%. Despite this similarity, the power limit of 80% (Experiment 7) resulted in a test time that is 46% smaller than the solution for a 50% power limit (Experiment 9), for the same number of pins and very similar area overheads. The limit of 50% is the minimum power limit for this system, which corresponds to the power consumption of Module 2 during test.

4.7.3 Benchmarks u226 and d281

In this section, the systems u226 and d281 are considered. The particularity of these two benchmarks is the presence of some modules that do not require a test access mech-

Table 4.12: Test requirements of benchmark h953

Module	Nb. of Scan chains	Scan chain depth	Nb. of I/Os	I/O test pins	Nb. of test patterns
0	0	0	12/41	0/0	0
1	4	348	112/152	5/5	341
2	2	327	68/89	3/3	9
3	2	32	9/17	3/3	39
4	4	21	88/67	9/9	49
5	4	121	19/13	5/5	110
6	4	185	15/11	5/5	182
7	0	0	80/32	80/32	65
8	8	189	35/69	9/9	305

Table 4.13: Test results for benchmark h953

Experiment	Optimized factors	No. of extra pins	Test time	Area overhead
1	PT	10	119,357	2%
2	PT	2	214,062	2.3%
3	PT	50	214,062	1%
4	PTA	21	119,357	0%
5	PTA	2	223,139	0%
6	PTA	3	138,442	1%
7	PTAP (80%)	3	148,966	0.5%
8	PTAP (50%)	19	123,320	1%
9	PTAP (50%)	3	218,391	0.4%
10	PTAP (50%)	9	135,123	0%

Optimization factors: PT = pins&time,
PTA = pins&time&area, PTAP = pins&time&area&power

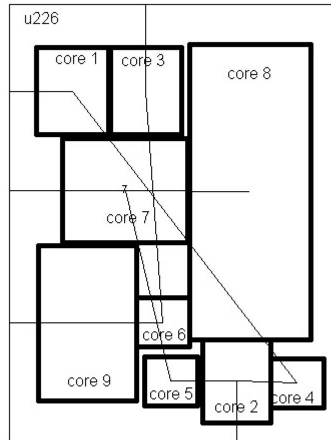


Figure 4.18: Floorplanning of benchmark u226

anism, that is, BISTed modules. The BIST testing can be either the single testing method of the module, as in system u226, or one of the tests defined for a core in addition to an external testing, for example, as in system d281. Both systems present only one level of hierarchy and do not define any system level testing (UDL test).

To model a BIST testing in the proposed test planning approach, the BISTed core is modeled as two CUTs: the first one represents the real core, with associated area, connections, and location in the chip. If the BISTed core also requires a TAM for another test, the test requirements of the external test are associated to this first CUT. The second CUT is a virtual core with no area. This virtual block contains the BIST test requirements. Usually, such a block requires one input pin (BIST enable) and one or more output pins (BIST-ready signal or signature). In addition, the power consumption of the core when a BIST test is being performed is assumed to be higher than the consumption of an external test. This is modeled in the tool by increasing the switching rate of the gates and flip-flops of the CUT modeling the BIST test. Although this CUT is not actually present in the system, it has the same location and connections of the actual CUT, so that a TAM for the BIST signals can be properly defined. With this model, the BIST test can be scheduled using the same procedure of the external testing, following the same priority rules of the other cores. The only difference is that the BIST testing time is fixed.

One can generalize the modeling of the BIST test to model a core with more than one test. A core can have a number of tests that require a TAM but must be scheduled independently. By defining a virtual CUT in the tool for each test, this situation can be easily and correctly modeled.

Benchmark u226 was firstly used in (COTA et al., 2002). The system is a small but complex system to be tested, because of the several types of tests related to the presence of both, analog and digital cores, in the same description. This benchmark was the first contribution to the ITC'02 set considering BISTed cores. Since this system is a contribution of this University, the connections among the cores are known. For this first experiment, the actual connections are used, and a functional bus is defined to connect some of the cores. The floorplanning of this benchmark is shown in Figure 4.18.

Table 4.14 presents the test requirements of this system. In this example, the BISTed cores do not require a TAM and to not use scan chains. Hence, the number of test patterns associated to those cores actually informs their test time.

Table 4.15 presents the test results for system u226, where the impact of BISTed

Table 4.14: Test requirements of benchmark u226

Module	Type of test	Nb. of Scan chains	Scan chain depth	Nb. of I/Os	I/O test pins	Nb. of test patterns
0	-	0	0	12/41	0/0	0
1	BIST	0	0	1/1	1/1	1,363,968
2	BIST	0	0	1/1	1/1	1,363,968
3	BIST	0	0	1/1	1/1	1,363,968
4	external	0	0	3/17	3/17	2,666
5	external	0	0	3/17	3/17	2,666
6	external	0	0	3/17	3/17	2,666
7	external	20	52	97/64	22/22	76
8	BIST	0	0	1/1	1/1	1,048,576
9	external	0	0	17/10	17/10	15

Table 4.15: Test results for benchmark u226

Experiment	Optimized factors	No. of extra pins	Test time	Area overhead
1	PT	1	1,363,968	9e-7%
2	PTA	3	1,363,968	1e-7%
4	PTAP (80%)	3	1,363,968	1e-7%
5	PTAP (50%)	3	1,460,140	1e-7%
6	PTAP (35%)	3	1,363,968	1e-7%

Optimization factors: PT = pins&time,
PTA = pins&time&area, PTAP = pins&time&area&power

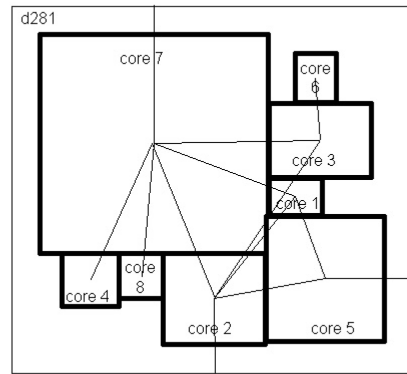


Figure 4.19: Floorplanning of benchmark d281

Table 4.16: Test requirements of benchmark d281

Module	Nb. of Scan chains	Scan chain depth	Nb. of I/Os	I/O test pins	Nb. of test patterns	
					External	BIST
0	0	0	190/176	0/0	0	0
1	0	0	60/26	60/26	26	256
2	0	0	233/140	233/140	158	2,048
3	0	0	207/108	2/2	96	2,048
4	4	8	45/52	11/11	90	256
5	6	32	214/228	14/14	118	256
6	2	9	32/32	6/6	80	256
7	20	32	700/790	45/45	0	2,048
8	2	9	32/32	6/6	58	1,024

cores can be verified. One can observe that the test time for the system does not change, for most system configurations. This test time is due to the BIST testing of one of the embedded cores, and determines the system test time. However, when a power limit of 50% is considered, there is an increase of 7% in test time. The minimum power limit of 41%, set by Module 7 changes the search space and the test time is again defined by the BISTed core. Additional results for this system will be presented in Section 4.8, when the impact of the random functional association among cores is evaluated.

The connections and floorplanning of the benchmark d281 are shown in Figure 4.19. Similarly to the benchmark d695, this system is also composed by a set of ISCAS'85 and ISCAS'89 circuits as cores. However, seven out of the eight cores have two tests: one that requires an access mechanism and another one that is autonomous (BIST test).

The number of test patterns per core for each test, as well as other test requirements of this system are shown in Table 4.16. In the table, the number of test patterns for each test (external and BIST) is presented in the last two columns but the test pins requirement (fifth column in Table 4.16) is only related to the external testing, since the BIST is supposed to be autonomous. For this example, the test time of the BIST test is defined during the scheduling, considering the length of the original scan chains of each module (third column in Table 4.16).

Table 4.17 presents the resulting test costs for the benchmark d281 presented in Figure 4.19. One can observe that the best solutions for time (Experiments 1 and 4) require

Table 4.17: Test results for benchmark d281

Experiment	Optimized factors	No. of extra pins	Test time	Area overhead
1	PT	155	3,926	38%
2	PT	1	8,300	24%
3	PT	8	26,691	5%
4	PTA	209	3,926	2%
5	PTA	2	30,784	2.5%
6	PTA	8	11,961	0%
7	PTAP (80%)	2	30,874	2.5%
8	PTAP (50%)	6	12,320	1.4%
9	PTAP (30%)	26	8,132	2%
10	PTAP (30%)	16	10,396	16%

Optimization factors: PT = pins&time,
PTA = pins&time&area, PTAP = pins&time&area&power

an excessive number of pins. However, a good compromise is found in Experiments 6 and 8.

4.7.4 Benchmarks p22810, p34392, p93791, and a586710

In this section, the systems with more than one level of hierarchy are presented. These systems present some super-cores, that is, cores that embed other cores. When a core is described as part of another module, it is assumed that one can access the core interface through the wrapper of the corresponding super-core. Therefore, the test of these two modules is described separately.

To model a super-core, one can use the same procedure defined to model the BISTed modules. Indeed, if one consider that the location of the super-core and its components is approximately the same, and that the embedded core usually connects to the rest of the system through the wrapper of the super-core, one can define the test of the embedded cores as additional tests of the super-core. Thus, for each module embedded into another core, a virtual CUT is created in the description of the system in the test planning tool. The virtual CUT has the same location and connections of the super-core of highest level (level 1) so that a TAM can be defined through the system connections. However, in this case, the area of the virtual core is considered, as well as its specific test requirements and characteristics, such as power consumption during test. With this model, the test of the super-core can be scheduled independently of the test of its components.

System p22810 is shown in Figure 4.20. It contains a total of 28 modules, and two of them (Modules 1 and 2 in the figure) are super cores. The connections among cores were defined in such a way that bidirectional pins at the modules interfaces are connected through two functional buses that are further connected to the bidirectional pins at the system interface.

Table 4.18 presents the test requirements of the cores in this system. The second column in this table indicates which level of the system hierarchy each core is located in. All cores have only one test, but there are two tests defined for the system-level (Module 0) logic. These UDL tests are also modeled as virtual cores to be inserted into the test scheduling.

Table 4.18: Test requirements of benchmark p22810

Module	Level	Nb. of Scan chains	Scan chain depth	Nb. of I/Os	I/O test pins	Nb. of test patterns
0	0	0	0	10/67/96	10/67/96	99
1	1	10	130	28/56/32	12/12	785
2	2	0	0	47/33	47/33	12,324
3	2	0	0	38/26	38/26	3108
4	2	0	0	48/64	48/64	222
5	1	29	214	90/112/32	31/31	202
6	2	0	154	80/64	80/64	712
7	2	0	514	84/64	84/64	2,632
8	2	0	1,473	36/16	36/16	2,608
9	1	24	122	116/123/32	25/25	175
10	1	4	99	50/30	4/4	38
11	1	8	88	56/23/71	10/10	94
12	1	11	82	40/23/71	13/13	93
13	1	4	104	68/149	5/5	1
14	1	3	73	22/15	2/2	108
15	1	6	80	84/42/32	9/9	37
16	1	1	109	13/43/72	3/3	8
17	1	4	89	223/69/32	6/6	25
18	1	5	68	53/11/32	7/7	644
19	1	3	43	38/29	4/4	58
20	1	4	77	45/40/2	5/5	124
21	1	10	186	115/76/64	9/9	465
22	1	3	77	54/40	4/4	59
23	1	7	115	31/8/35	5/5	40
24	1	5	101	73/23/35	2/2	27
25	1	18	181	58/46/86	20/20	215
26	1	31	400	66/33/98	33/33	181
27	1	1	34	285/94	10/10	2
28	1	5	100	48/43	6/6	26

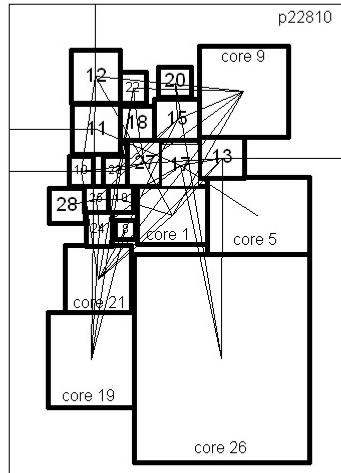


Figure 4.20: Floorplanning of benchmark p22810

Table 4.19: Test results for benchmark p22810

Experiment	Optimized factors	No. of extra pins	Test time	Area overhead
1	PT	307	102,965	7%
2	PT	102	239,946	3.9%
3	PT	188	191,111	3%
4	PTA	282	102,965	2%
5	PTA	76	281,626	1%
6	PTA	130	279,551	0%
7	PTAP(80%)	76	281,626	1%
8	PTAP (50%)	67	385,389	1%
9	PTAP (30%)	78	290,398	2%

Optimization factors: PT = pins&time,
PTA = pins&time&area, PTAP = pins&time&area&power

Table 4.19 presents the test solutions found for this system considering the connections and floorplanning shown in Figure 4.20. Again, all solutions present a number of extra pins quite high, compared to the number of functional pins of the system, and the best solution for pins (Experiment 8) presents the highest test time among the selected solutions. On the other hand, a compromise is found in Experiments 5 an 7. Moreover, this compromise is kept for a power limit as low as 30%.

Figure 4.21 shows the connections and floorplanning defined for the benchmark p34392. This benchmark contains 19 modules divided into four super-cores, and no functional bus is assumed in the interconnections.

Table 4.20 presents the test requirements of the cores in this system. All cores have only one test, and there are two tests defined for the system-level (module 0) logic. Table 4.21 presents some test solutions found for this system.

For this benchmark, the best solution for time considering only pins and time optimization, is also the best solution for area under those constraints (Experiment 1). One can notice that the best solutions for pins (Experiments 2, 6, 8, and 10) present a test time that is more than three times the minimum time for the system. The best trade-off for this

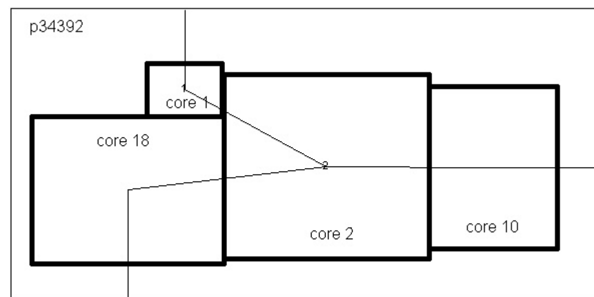


Figure 4.21: Floorplanning of benchmark p34392

Table 4.20: Test requirements of benchmark p34392

Module	Level	Nb. of Scan chains	Scan chain depth	Nb. of I/Os	I/O test pins	Nb. of test patterns
0	0	0	0	32/27/114	32/27/114	27
1	1	1	806	15/94	2/2	210
2	1	29	570	165/263	17/17	514
3	2	0	0	37/25	37/25	3,108
4	2	0	0	38/25	38/25	6,180
5	2	0	0	62/25	62/25	12,336
6	2	0	0	11/8	11/8	1,965
7	2	0	0	9/8	9/8	512
8	2	0	0	46/17	46/17	9,930
9	2	0	0	41/33	41/33	228
10	1	19	519	129/207	12/12	454
11	2	0	0	23/8	23/8	9,285
12	2	0	0	7/4	7/4	173
13	2	0	0	12/16	12/16	2,560
14	2	0	0	11/8	11/8	432
15	2	0	0	22/8	22/8	4,440
16	2	0	0	7/7	7/7	128
17	2	0	0	15/4	15/4	786
18	1	14	729	175/212	11/11	745
19	2	0	0	62/25	4/4	12,336

Table 4.21: Test results for benchmark p34392

Experiment	**Optimized factors**	**No. of extra pins**	**Test time**	**Area overhead**
1	PT	91	544,579	1%
2	PT	2	1,482,169	7.9%
4	PTA	79	544,579	1%
6	PTA	2	1,737,079	4%
7	PTA	19	814,378	1%
8	PTAP (80%)	2	1,737,079	4%
9	PTAP (50%)	56	764,771	1%
10	PTAP (50%)	2	1,802,461	2%
12	PTAP (50%)	26	897,662	4%
13	PTAP (30%)	386	1,075,242	1%
14	PTAP (30%)	49	1,368,792	0.76%

Optimization factors: PT = pins&time,
PTA = pins&time&area, PTAP = pins&time&area&power

system is found in Experiment 7, when only pins, time, and area are optimized. However, when power is also considered, a similar test time can be found (Experiment 12) at the expense of the pin and area overhead.

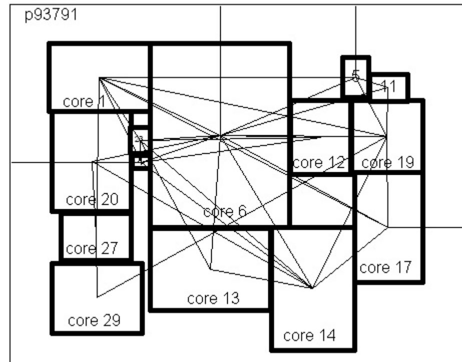


Figure 4.22: Floorplanning of benchmark p93791

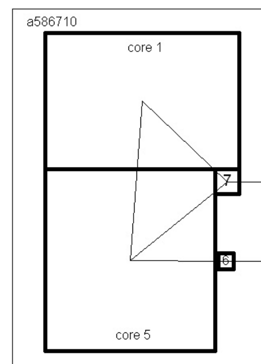


Figure 4.23: Floorplanning of benchmark a586710

System p93791 is very similar to the two previous benchmarks in terms of test requirements. There are no BISTed cores, each core has a single test, and there is no system-level test defined. For this benchmark, a functional bus is also assumed to connect the bidirectional pins in the interface of embedded cores.

Figure 4.22 shows the floorplanning of system p93791, Table 4.22 summarizes the test requirements of the embedded modules, and Table 4.23 presents the test solutions found with the proposed test planning tool.

One can observe that the area overhead for this system is high in all solutions. It is important to notice, at this point, that the area occupied by the inserted test resources is evaluated as an absolute value. This means that the distance among cores, the area of basic gates (flip-flops, multiplexers), and the width of the wires, for instance, are defined considering an specific technology. The area of the cores is also estimated considering the number of flip-flops and gates for the same technology. However, the relationship between these two areas (the original area of the system and the additional area for testing) that gives the area overhead estimation, may not represent the reality. However, in spite of this factor, the variation in area overhead in Table 4.23 demonstrates the exploration of the design space for that benchmark.

In addition, one can observe that the best solution for time (Experiments 1 and 4) is actually unfeasible, because of the pin and area overhead. The best trade-offs are given in Experiments 5 and 6, although the number of pins in both solutions can be considered high.

System a586710 floorplanning is shown in Figure 4.23. This benchmark contains one super-core (Core 1 in the figure) that embeds two BISTed modules. Table 4.24 summarizes the test requirements of this system, and Table 4.25 presents some test solutions

Table 4.22: Test requirements of benchmark p93791

Module	Level	Nb. of Scan chains	Scan chain depth	Nb. of I/Os	I/O test pins	Nb. of test patterns
0	0	0	0	103/79/66	103/79/66	0
1	1	46	168	109/32/72	48/48	409
2	2	0	0	40/34	40/34	192
3	2	0	0	40/29	40/29	648
4	1	23	5	15/30/72	47/47	11
5	1	0	0	102/80/66	102/80/66	6,127
6	1	46	521	417/324/72	49/49	218
7	2	0	0	9/32	9/32	177
8	2	0	0	9/32	9/32	177
9	2	0	0	43/34	43/34	192
10	1	0	0	267/128	267/128	1,164
11	1	11	82	146/68/72	11/11	187
12	1	46	93	289/8/72	52/52	391
13	1	46	219	111/31/72	49/49	194
14	1	46	219	111/31/72	49/49	194
15	2	0	0	44/34	44/34	288
16	2	0	0	137/64	137/64	396
17	1	43	150	144/67/72	46/46	216
18	2	0	0	79/34	79/34	42
19	1	44	100	466/365/72	54/54	210
20	1	44	181	136/12/72	47/47	416
21	2	0	0	79/34	79/34	42
22	2	0	0	42/34	42/34	42
23	1	46	175	105/28/72	48/48	234
24	2	0	0	17/4	17/4	3,072
25	2	0	0	29/16	29/16	2,688
26	2	0	0	42/34	42/34	96
27	1	46	68	30/7/72	50/50	916
28	2	0	0	109/50	109/50	396
29	1	35	189	117/42/72	38/38	172
30	2	0	0	43/334	43/334	192
31	2	0	0	148/70	148/70	204
32	2	0	0	268/128	268/128	3,084

Table 4.23: Test results for benchmark p93791

Experiment	Optimized factors	No. of pins	Test time	Area overhead
1	PT	1,080	114,317	174%
2	PT	120	607,400	149%
3	PT	287	611,135	109%
4	PTA	1,291	114,317	28%
5	PTA	77	530,667	22%
6	PTA	153	541,723	12%
7	PTAP (80%)	130	523,045	31%
8	PTAP (50%)	100	586,738	27%
9	PTAP (30%)	11	1,615,500	23%

Optimization factors: PT = pins&time,
PTA = pins&time&area, PTAP = pins&time&area&power

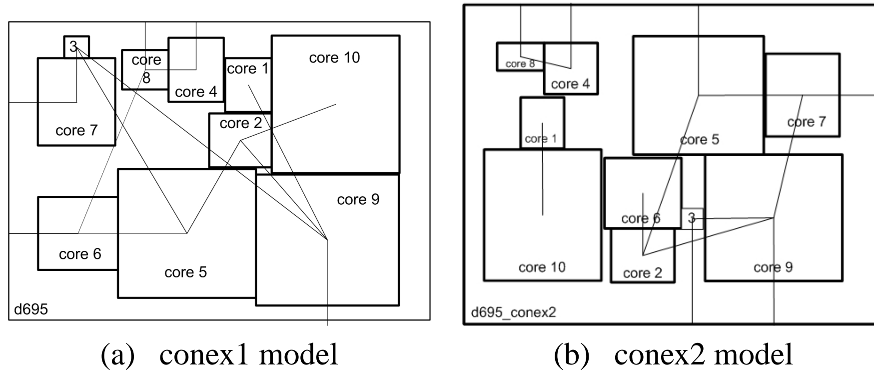
Table 4.24: Test requirements of benchmark a586710

Module	Level	Nb. of Scan chains	Scan chain depth	Nb. of I/Os	I/O test pins	Nb. of test patterns
0	0	0	0	31/59/111	31/59/111	0
1	1	8	2,155	437/370	10/10	2,945
2	2	0	0	275/222	1/1	2,679,692
3	2	0	0	407/244	1/1	6,029,308
4	2	0	0	206/324	206/324	181,140
5	1	8	2,548	343/218/111	10/10	2,945
6	1	0	0	34/35	34/35	40,431
7	1	0	0	226/100	9/32	1,914,433

Table 4.25: Test results for benchmark a586710

Experiment	Optimized factors	No. of pins	Test time	Area overhead
1	PT	294	7,739,141	1%
2	PT	3	142,130,961	3.5%
3	PT	10	29,820,523	0%
4	PTA	297	7,739,141	1%
5	PTA	12	77,364,905	0.25%
6	PTA	17	15,726,859	0%
7	PTAP (80%)	11	29,811,685	0.05%
8	PTAP (50%)	30	22,069,597	0.4%
9	PTAP (30%)	40	25,414,419	0.5%

Optimization factors: PT = pins&time,
PTA = pins&time&area, PTAP = pins&time&area&power



(a) conex1 model (b) conex2 model

Figure 4.24: Two configurations for system d695

found with the proposed test planning tool. For this benchmark, the best solution is given in Experiment 6, although the test time for this solution is 100% higher than the minimum test time for the system. The number of test resources in this system is very limited, which contrasts with the large number of pins and test patterns of the embedded cores.

4.8 System Characteristics and Benchmark format

In Section 4.7, the ITC'02 benchmarks were used to validate the proposed test planning approach. As explained earlier, all the system information necessary to model those systems in the ReBaTe tool (connections among cores, floorplanning, power consumption of the cores, etc) were estimated, since the benchmarks format describes only the system test requirements. Therefore, the test solutions presented for those systems are valid for the estimated system information. However, several other estimations of the system information can be defined for the ITC'02 benchmarks. For example, one can easily implement another algorithm to define the interconnections among cores, creating a completely different system based on the same set of cores. For this new system, the test solution devised by the ReBaTe tool would be very different from the results presented until now. The impact of the extra system information on the test solution results is thus discussed in this section using systems d695, p22810, p93791, and u226 as example.

For the system d695, a second functional association among cores and its corresponding floorplanning was defined. Figure 4.24(a) represents the first connection model shown in Figure 4.12 of Section 4.7.1, and Figure 4.24(b) shows the new floorplanning defined for the second connection model. These two floorplannings are referred here as *d695_conex1* and *d695_conex2*, respectively.

The test results for this system are presented in Table 4.26. For each functional connection model, two test results are presented: the first one where pins, test time, and area (PTA) are optimized, and a second one, where only pins and test time (PT) are optimized.

Table 4.26 confirms that the proposed method is very dependent on the system characteristics. In the second connection model (*d695_conex2*) there are more unconnected output pins and higher fan-in (reduced bitwidth of functional connections) for the cores, although the same number of functional system pins is kept. One can see that the new connections led to worse solutions if compared to the original ones, with larger bitwidth between cores and more pins connected.

From the first two experiments of Table 4.26, one can remark how the location and communication of the cores inside the system can interfere in the test solution. If only the

Table 4.26: New test planning results for d695

Experiment	**Test Time (cycles)**	**No. of extra Pins**	**Area Overhead**
d695_conex1 (PT)	21,986	0	18%
d695_conex2 (PT)	46.627	20	21.4%
d695_conex1 (PTA)	37,089	3	3.8%
d695_conex2 (PTA)	43,488	2	7.6%

Optimization factors: PT = pins&time, PTA = pins&time&area

Table 4.27: New test planning results for p22810

Experiment	**Test time (cycles)**	**No. of extra Pins**	**Area Overhead**
WB-conex1	347.919	84	0.93%
WB-conex2	785.423	46	0.31%
NB-conex3	1.592.739	51	0.29%
WB-conex4	2.059.441	8	0.15%

Optimization factors: PTA = pins&time&area

number of test pins in the interface is used to guide the definition of the global test solution, it is very likely that two systems with similar sets of cores but with distinct layouts will present very similar test solutions. However, one can observe that very dissimilar solutions can be found regarding both, the test time and the resulting area overhead, when multi-variable optimization is sought and system information is used. Therefore, the exploration of the system design space before considering the inclusion of test pins and test buses may be worth. Moreover, one can observe that the less optimization factors are considered in the solution, the better the test time, at the cost of the area overhead.

For the benchmark p22810, the existence of bidirectional pins opens more possibilities for the test solution. Four experiments were performed for this system and are presented in Table 4.27: two assignments for functional connections and respective placement considering the existence of buses connecting the bidirectional pins (experiments *WB-conex1* and *WB-conex2*, respectively), another assignment when no buses were assumed in the design (experiment *NB-conex3*), and a different functional assignment for the same placement definition of experiment *WB-conex1*, also using a functional bus (experiment *WB-conex4*). The first functional connection considered in Table 4.27 (*WB-conex1*) is the model presented in Section 4.7.4. The last experiment (*WB-conex4*) shows whether the placement information can be dissociated from the functional connection definition, that is, if it can be neglected during test synthesis when functional connections are known. In all cases, the optimization factors are the number of pins, the test time, and the area overhead.

Comparing the experiments *WB-conex1* and *WB-conex2* in Table 4.27, one can notice again the impact of system level information in the test planning solution, since the same variability of results from Table 4.26 is present. Furthermore, comparing the results of experiments *WB-conex1* and *WB-conex2* with the results of experiment *NB-conex3*, it becomes clear that the assumption of the existence of an internal bus in the system completely modifies the test result, when reuse of system hardware is assumed.

Table 4.28: New test planning results for p93791

Experiment	Test time (cycles)	No. of extra Pins	Area Overhead
WB-conex1	136.176	4	2.96e-3%
NB-conex2	104.168	0	1.63e-4%

Optimization factors: PTA = pins&time&area

Table 4.29: New test planning results for u226

Experiment	Test time (cycles)	No. of extra Pins	Area Overhead
NB-conex1	136,176	4	2.96e-3%
WB-conex2	104,168	3	1e-7%
NB-conex2	128,142	0	5.63e-3%

Optimization factors: PTA = pins&time&area

Finally, experiments *WB-conex1* and *WB-conex4* show that no system information can be neglected during test synthesis if multi-variable optimization is being performed. Since we have the same floorplanning but different connections in these experiments, the obtained solution can be very distant from the solution given for possibly more realistic models, as the ones shown in experiments *WB-conex1* and *WB-conex2*, for example. The increase in the test time for the experiment *WB-conex4* occurs as a response to the high area and pin overhead caused by the floorplaning that is not related to the functional connections.

For system p93791, the original connection model that considers a functional bus is compared to a new model where no functional bus is assumed. The test results for these two configurations is shown in Table 4.28 for pins, test time, and area optimization.

Table 4.29 shows the results for the system u226. For this benchmark, one can compare the test results for the random functional association to the results devised when the actual connections among cores is used. Thus, one can measure how the lack of the functional connection information can affect the test solution in a systemic approach.

It is important to notice that this system presents four cores with internal BIST structures that perform their test independently from the system resources. In fact, the BIST facility defines the total test time for this system, which is 1.363.968 cycles in all experiments of Table 4.29. Thus, the test time shown in Table 4.29 represents the test time only for those cores in u226 that actually use the defined TAM. Moreover, from Table 4.29 one can observe how far a test solution can be from the real result when the system level information is estimated. Again, pins, test time and area overhead are being optimized at the same time.

The first experiment in Table 4.29 (*NB-conex1*) represents the random assignment of the functional connections, with an associated floorplanning, for system u226. The last two experiments show the test solution for the actual functional connections of this system. In *WB-conex2*, an internal bus is assumed, while in *NB-conex2* core-to-core connections are used. Notice that the result for the estimated system is worse not only in terms of pin overhead, but also in test time, while the area overhead is one order of magnitude greater than the real system using a functional bus. Comparing the results of

Table 4.29, one can clearly see that the solution considering an assumed, rather than the real system information, is considerably different from the solution that can actually be found in a real situation.

Based on the results shown in this section, it becomes clear that system information does have an impact on test synthesis when such variables are taken into consideration. In other words, if the test planning task is performed along with the system synthesis, rather than being an isolated task, the space for global optimizations is largely expanded.

The current ITC'02 SOC Test benchmarks clearly represent the complexity of the test of core-based systems. They also point out the difficulties for the system integrator to implement a global test solution with restricted information about the embedded cores. Besides, the benchmarks set can also be used to stress the difficulty to implement test planning solutions in earlier design steps. However, the use of the current set for the comparison among test planning approaches must be carefully considered, whenever system information is used by a test planning tool. As shown by the experimental results of this section, this comparison can only be done if the same assumptions about the extra system information are considered in all methods.

On the other hand, it is very difficult to have this extra information, such as placement and functional connections data in a publicly accessible example of a system, since IP protection also for the system must be considered. However, it would be interesting, for example, to verify the possibility of including a common definition of all extra information (even if not real) for the set of benchmarks. This way, although not strictly precise, the comparison of different approaches could be more easily developed, enriching even more the quality and usage of the benchmarks.

5 NOC-BASED TESTING OF CORE-BASED SYSTEMS-ON-CHIP

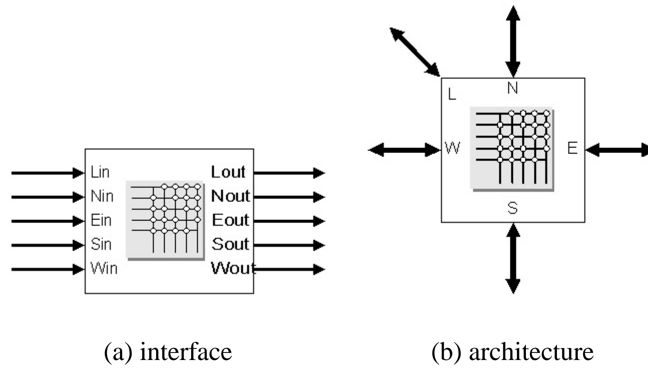
If a network-on-chip (NoC) is the communication platform of the system, the electronic access to each embedded core is available, since there is a real connection among all cores within the chip. The idea of reusing this resource during test is straightforward, since the access to the embedded cores is one of the main problems of the SoC test. If this reuse is possible, the pin and area overhead caused by the testing structures can be strongly reduced. The main remaining problem is, thus, the test time.

In this chapter, the impact of the reuse of a NoC for the test of core-based systems is discussed. A reuse strategy aiming at minimizing the system test costs is formalized in Section 5.1. The experimental results presented in Section 5.3 show that very reduced test times can be achieved, making the NoC reuse a very cost-effective test access mechanism. The proposed strategy is further evaluated in Section 5.4, with respect to a number of system configurations: different placements of the cores in the network, different number of interfaces with the tester, different network topologies, and test under power consumption constraints.

The main characteristics of a NoC-based design are described in Section 2.2. A packet-switched network model named SOCIN (System-on-Chip Interconnection Network), developed at UFRGS (ZEFERINO; SUSIN, 2003), is used in the experimental results. Each router in the SOCIN network is composed of five input and five output ports, as shown in Figure 5.1(a). One pair of ports is dedicated to the connection of the router with the core, while the remaining four pairs connect four communication channels around the router, as depicted in Figure 5.1(b). A SOCIN router is implemented using from 3,000 to 6,000 gates, depending on the bitwidth of the network channel and depth of the input buffers (ZEFERINO; SUSIN, 2003).

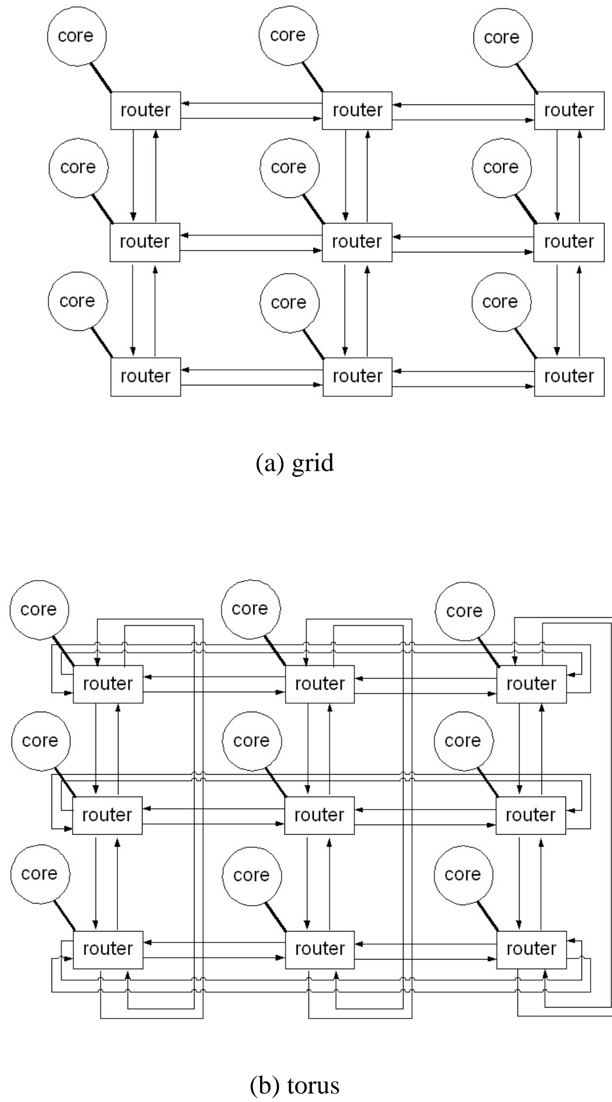
Two bi-dimensional network topologies were used in the experiments presented in this dissertation: torus and grid. Figure 5.2 shows these two topologies. The channels were defined to be 16-bit or 32-bit wide, with packets having unlimited length.

SOCIN uses credit-based flow-control and XY routing - a deadlock-free, deterministic and source-based approach, in which a packet is firstly routed on the X direction, and after on the Y direction before reaching its destination. Switching is based on the wormhole approach, where a packet is broken up into flits (flow control units), the smallest unit over which the flow control is performed, and the flits follow the header in a pipeline way. The flit size equals the channel width, and routers include a 4-flit queue at each input port. Figure 5.3 shows the implementation of the benchmark d695 in the grid topology of the SOCIN network.



(a) interface (b) architecture

Figure 5.1: Basic structure of the SOCIN router (ZEFERINO; SUSIN, 2003)



(a) grid

(b) torus

Figure 5.2: SOCIN topologies

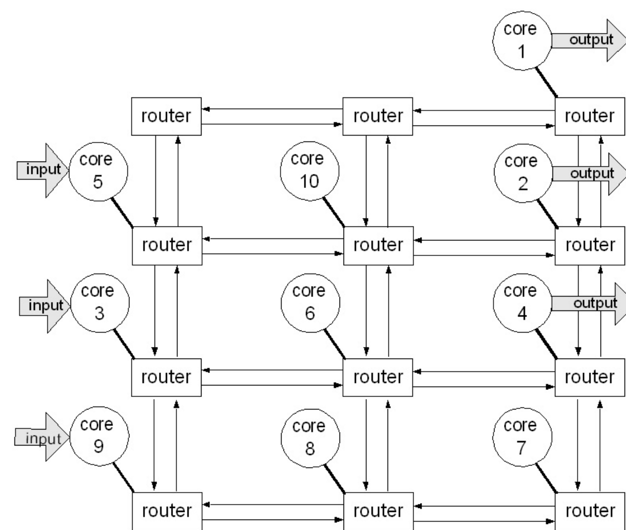


Figure 5.3: System d695 implemented in a 4x3 grid SOCIN NoC

5.1 Using the NoC During Test

In order to reuse the network-on-chip as the test access mechanism for the system cores, the test vectors and test responses of each core must be expressed as a set of packets to be transmitted throughout the network. Then, the wrapper that connects the core to the network must be modified to correctly send and receive the test data to/from the test interface of the core (scan controls, scan pins, functional pins).

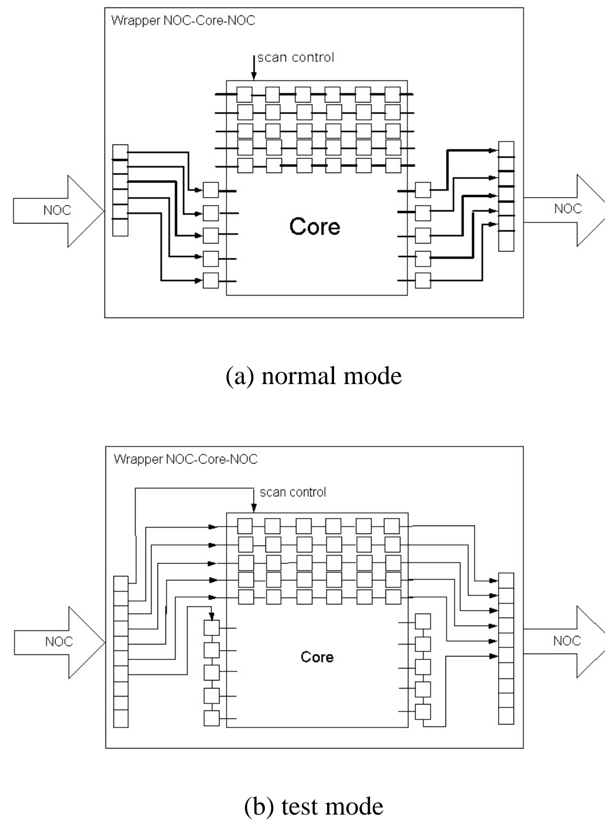
To keep the wrapper of each core as close to the original design (defined according to the application implemented in the network) as possible, the test packets are defined so that each flit arriving from the network is unpacked in one cycle. This means that each bit of a packet flit fills exactly one bit of the defined scan chains of the core. Functional inputs and outputs of the core, as well as the internal scan chains, are concatenated into external scan chains of similar length, in such a way that the channel width is enough to transport one bit of each scan chain. This process is very similar to the pre-processing of the cores defined in Section 4.1. One can assume that the test pins defined as shown for the ReBaTe tool are further concatenate until the total number of test pins in each direction fits the channel width.

Figure 5.4(a) shows an example of a wrapper during normal operation, and Figure 5.4(b) shows this wrapper modified to be reused during test. The area overhead due to the implementation of the test mode in the wrapper is comparable to the overhead of a basic P1500 wrapper. Basic wrapper cells containing scan flip-flops as the ones proposed in (MARINISSEN; KAPUR; ZORIAN, 2000) are required to implement the boundary scan chains with the functional pins at the core interface and to concatenate the original scan chains of the core when necessary. Then, the control signal to those cells are included into the input and output control of the original wrapper. Finally, two registers and a counter are required to store the address of the output interface for the test response and the delivery time of the response packet, respectively.

Notice that other configurations for the test packet could be used, such as more than one vector per packet, for instance. However, additional modifications in the wrapper to implement the test mode would be required. Hence, for this initial approach, the simplest packet configuration is being considered.

Control information, such as scan shift and capture signals, for example, are also delivered in the form of packets, either in the test header (to be interpreted by the wrapper) or as specific bits in the payload (for direct connection to the target pins). In Figure 5.4(b), the latter is assumed. In both cases, a test enable signal in the packet header indicates to the wrapper that a test configuration must take place. Furthermore, the original buffer structure of the router, designed according to the functional requirements, is re-used as it is, to reduce area overhead.

For the example d695, the data used to define the test packets is presented in Table 5.1. For each core, the number of test packets and the number of test pins at the core interface are shown in columns two and three, respectively. The number of packets to be transmitted is twice the number of test vectors (one for the vector and another one for the test response). Each packet comprises all bits of the test vector or the test response, divided into a number of flits corresponding to the scan chain of maximum length defined for the core, as explained above. The remaining three columns in Table 5.1 present the number of bits per packet, and the resulting number of flits of each packet considering communication channels of 32 and 16 bits, respectively. Usually, for scan-based cores, the numbers of flits for the test vector and for the test response are the same. For non scan-based cores, these numbers may be different, since they only depend on the number



(a) normal mode

(b) test mode

Figure 5.4: Wrapper configurations

of functional inputs and outputs of the core. Notice also that for some cores in Table 5.1 (3, 4, and 8), the same number of flits per packet is used for both channel widths. This happens because the original number of scan chains combined with the functional inputs and outputs of those cores is smaller than 16. For instance, let us define the number of flits per packet of Core 5, considering the physical channel of 32 bits. As shown in Table 4.2, this core has 32 scan chains of 45 bits each, 38 input pins, 304 output pins, and 110 test patterns. The number of test pins initially defined for this core is 39 ($32 + \lceil\frac{304}{45}\rceil$). However, the maximum number of test pins for the 32-bit communication channel is 32. Hence, the defined scan chains must be concatenated until this limit is achieved. Applying the same reasoning, the number of test pins (39) is divided by the number of available bits (32), giving the number of chains that must be concatenated (2). Then, concatenating the chains in groups of two, the length of the new chain is 90 bits (45*2). Since each bit of the chain must be loaded by one flit of the payload, the number of flits per packet for this core is 90.

One can consider that an external tester is connected to the functional system interface. Thus, the input and the output ports of the network can be re-used for the transmission of all packets (vectors and responses) to/from all cores. Let us also consider that only the cores and their corresponding wrappers are put into test mode, while the network routers and channels are kept in normal mode. This way, the same protocol used for functional communication is used during test. Notice that the test is still off-line, since the cores are in test mode. Moreover, although the test of the network structures in conjunction with the test of the embedded cores is being currently studied, this issue is not addressed in

Table 5.1: Test packets for system d695

Core	# of packets	# Test pins	Pack. size (bits)	Flits/Pack. 32-bit	Flits/Pack. 16-bit
1	24	64	32	1	2
2	146	315	108	5	7
3	150	36	33	32	32
4	210	79	250	54	54
5	220	374	1730	90	179
6	468	230	790	41	80
7	190	243	684	34	67
8	194	88	228	46	46
9	24	387	2048	108	216
10	136	166	1742	102	204

this work. It is assumed that the network is tested in a previous step and is fault-free.

The number of system interfaces used during test defines the initial number of paths that can be used in parallel to transmit test data. However, if there is more than one interface between the system and the external tester, the order on which the cores are tested is important, since different paths (and, consequently, different conflicts over the network resources) will be used for each core depending on the system input/output available at that time.

In addition, the input where the test vectors are delivered from and the output where the responses are sent to, determine the core test time. For example, if the input ports of the system shown in Figure 5.3 are Cores 5 and 9, the input access paths for Core 6 are $5 \rightarrow 10 \rightarrow 6$ and $9 \rightarrow 8 \rightarrow 6$, in the SOCIN network. If the output cores are located at Cores 2 and 4, the possible output access paths for Core 6 are $6 \rightarrow 4$ and $6 \rightarrow 4 \rightarrow 2$. The number of routers is the same for both input paths. However, for the output, the second path uses four routers, requiring more cycles to transmit each test packet than the first path with three routers.

One possibility of reuse is to define the test of the system as another application to be run in the network, that is, the tester could randomly choose the system interface to send the test patterns and the network controls the traffic of all test packets, dynamically solving the conflicts over the routers and channels. However, this approach can lead to a sub-utilization of the network resources. Indeed, the tester can choose an interface that leads to a blocked channel, delaying the message, whereas there are other paths within the network that are free to be used.

On the other hand, the communication requirements of the whole test application are known before its execution in the network, since the number and size of each package are previously defined. Besides, the routing and arbitration algorithms of the network are also known, that is, one can statically define the path that each packet can take in the network, so that all free channels in the network are used in parallel.

We propose a scheduling approach to systematize and automate the definition of test access paths through the network and a test sequence that optimizes the total test time. Considering the reuse of the functional system interfaces, the test costs in terms of pins can be drastically reduced. The area overhead caused by the modifications in the wrapper are also minimal, in the order of a basic P1500 wrapper. It is interesting to notice that

the bypass mode of a core is already implemented by the network router, and not by the wrapper that connects the core to the NoC. Therefore, there are only a few operation modes that must be added to the wrapper function. Basically, it needs to provide a test mode and a bypass mode in the case of the super-cores. Thus, if the test time is minimized by scheduling the test packets a priori, the proposed approach can be very efficient.

The network has the same functionality during the normal and test modes of the system, but it can operate, for example, at a lower frequency during test, according to the cores and tester requirements. Moreover, the placement of the cores over the routers as well as the number of interfaces with the tester are assumed to be previously defined by the application.

5.1.1 Exploiting pipeline within the NoC

With the scheduling of the test messages, the network arbitration will no longer be required during test, since all conflicts are solved statically, during the scheduling planning.

Let us consider the packets that must be transmitted through the network as the tasks to be scheduled, and the different access paths for each core as the resources that can be used by those tasks. Each task (packet) can use any available resource (path) capable of delivering the message to the correct destination. This means that different packets delivered to/from the same core can take different paths within the network. For the response packets (from the core to the tester), the test header of the packet contains its origin. Moreover, the packet carrying a test vector also carries the address of the output interface for the response packet of that vector. This information is defined during the scheduling and actually inserted into the test packet by the tester during test.

Equation 5.1 defines the time required to transmit a packet through a path. T_{router} indicates the number of cycles spent by the packet header in each router to establish the path; $Nb_{routers}$ is the number of routers in the path; $T_{headers}$ indicates the cycles required to pack and unpack the headers; $payload$ is the number of flits of the packet. For the SOCIN network, $T_{router} = 3$ and a test header is assumed, in addition to the packet header originally present in each message.

$$T_{packet} = T_{headers} + T_{router} * Nb_{routers} + payload \tag{5.1}$$

For each packet, two extra cycles are required by the core and its wrapper to process the test and be ready to pack and deliver the response packet.

There are two precedence rules for this problem. The first one is related to the test responses. A packet containing a test response can only be sent after the corresponding vector is completely received by the core and processed. In traditional scan-based test schemes, test time is minimized by performing parallel scan-in and scan-out operations. That is, while a test response is extracted from the scan chain of the core, a new test vector is being injected in the chain. In the network, this parallelization is only possible if an input payload (test vector) is received by the core at the same time this core delivers an output payload (test response). For this to happen, the new vector packet must be delivered by the tester several cycles before the core finishes the processing of the previous vector. Indeed, a new packet can be sent as early as an input path becomes available, as long as the response of the previous vector is not blocked. Thus, the scheduling of each packet must consider the possibility of conflicts and subsequent blocking of the output path, to avoid loss of data.

In the proposed approach, each channel in the network is assigned a time information, indicating when the channel is free to be used by a new packet. With this information, it

is possible to schedule the next vector packet of a core as soon as one path is available, provided that the packet will not arrive at the core interface before the response payload can proceed in the network. This strategy combines the network and the core parallelism while ensuring that the internal scan chains are not overwritten.

The second precedence rule is more general, and deals with the priority of use of a given path. One can define that cores with larger number of packets and larger size of packets have priority to use shorter paths to reduce test time. The idea is to associate the shortest path to the most expensive core, to minimize its test time. This rule is detailed in Section 5.1.3, when the chosen scheduling algorithm is explained.

Each packet of each core must be transmitted from the tester to the core and vice-versa. Furthermore, the original buffer structure of the router, designed according to the functional requirements, is re-used as is. The scheduling algorithm ensures the defined buffer is enough, since all conflicts are statically solved.

With these definitions, a variation of the list-scheduling algorithm can be implemented, and the proposed algorithm is explained in Section 5.1.3. In the next section we explain the modeling of the power consumption during test, which will also be used in the scheduling definition.

5.1.2 Power Consumption Calculation

The main advantage of the network reuse during test is the possibility of parallelization provided by this communication platform. However, as more cores are tested in parallel, the system power consumption during test may become an issue. Therefore, the scheduling algorithm must also consider power consumption.

In the proposed test technique, there are four sources of power consumption: the core, the wrapper, the router, and the communication channel.

Similarly to the calculation of the power consumption of a core presented in Chapter 4, Equation 5.2 gives the dynamic consumption per cycle of a network router for the transmission of a single packet. C_L is the load capacitance (technology-dependent constant), T is the clock period, and σ is the switching factor. Variables nb_{ff} and nb_{gt} represent the number of active flip-flops and gates in the router, respectively, when one packet is being routed, while σ_{ff} and σ_{gt} are the switching factors for flip-flops and gates, respectively. Notice that for the flip-flops there is a constant switching factor caused by the clock, in addition to the eventual switching in the stored bit value.

$$P_{router} = C_L * Vdd^2 * \frac{1}{T} * [(\sigma_{ff} + 1) * nb_{ff} + \sigma_{gt} * nb_{gt}] \quad (5.2)$$

Equation 5.3 evaluates the power consumption of a communication channel in the NoC structure. The load capacitance of the channel is given by the product of the number of wires in the channel (ch_w), the length of the channel (ch_l), and the width of the wire ($wire_w$). Variable σ_w is the switching factor for the wire. In this first approach, all channels are assumed to have the same length, although this may be not the case in the actual implementation of the communication platform. As the power consumption is calculated per cycle, the size of the packet being transmitted is not important at this point.

$$P_{channel} = C_L * Vdd^2 * \frac{1}{T} * \sigma_w * (ch_l * wire_w * ch_w) \quad (5.3)$$

The total power consumption for a packet transmission is calculated according to the path established in the network for that packet: for each router and each channel active in the path, the router and channel consumptions are added.

For example, let us consider a packet containing the test vector v for core 6 is delivered by the tester through the system input at core 5, in the network shown in Figure 5.3. Consider the power consumption of one router for the transmission of one packet as P_{router}, and the power consumption of the channel as $P_{channel}$. When Core 5 is the chosen system input, the path between Cores 5 and 6 includes the routers of Cores 5, 10, and 6, as well as the communication channels 5-10 and 10-6 (see Figure 5.3), considering the XY routing policy of the SOCIN network. Thus, the power consumption of this network transmission for one packet is calculated by Equation 5.4.

$$P_{packet}(v) = 3 * P_{router} + 2 * P_{channel} \tag{5.4}$$

Equation 5.4 can be generalized to calculate the power consumption for the transmission of each packet, as shown in Equation 5.5, for $nb_{routers}$ the number of routers and $nb_{channels}$ the number of channels in the path.

$$P_{packet} = nb_{routers} * P_{router} + nb_{channels} * P_{channel} \tag{5.5}$$

The power consumption of a core during test depends on the core logic, on the test vectors, and on the order of application of the patterns. We assume the power consumption of each core during test is known by the core provider. Moreover, as each wrapper is usually developed for a specific core, we also assume the wrapper consumption is provided by the core designer. For the experimental results presented in Section 5.3, the power consumption calculated in Chapter 4 for each module of the ITC'02 benchmarks will be reused.

Since each test vector is packed, transmitted, and processed separately, the peak power consumption for receiving (wrapper consumption) and processing (core consumption) a single test vector is considered, that is, the power consumption P_{core} assumed for each core and its wrapper corresponds to the power consumption of the test pattern with highest consumption for that core. Moreover, a consumption per cycle is considered, so that it becomes independent of the test frequency. In fact, any power profile for the cores can be used. If there is enough information for each core, a more accurate power profile can be defined and included in the proposed method. Also, it is important to notice that the proposed technique does not assume any type of manipulation of the test set in order to reduce the power consumption. The basic assumption is that the method of transmission of the test vectors (one per packet) already contributes for the power consumption minimization of the core, and the main goal is to define a power-constrained test scheduling algorithm for the on-chip network TAM. However, if there is a specific order of test application that further reduces the consumption of the core or of the network, it can be easily implemented by changing the initial order of the test patterns and establishing additional precedence rules in the test scheduling algorithm.

5.1.3 Power-Aware Test Scheduling

The scheduling can be modeled as a resource-constrained problem, since there is a limited number of resources (paths within the network) that can be used by the packets. Each packet of each core must be scheduled, to ensure that a response for each test vector is delivered. The goal is to minimize the total execution time of the tasks, and maximize the utilization of the available resources. For this model, a version of the list-scheduling algorithm can be used (GEREZ, 1998). The main idea is to process the available time instants in increasing order (starting at zero) and schedule as many operations as possible at a certain instant before moving to the next. Let us assume that a *ready-list* L_t contains

the packets that can be scheduled from a given instant of time, and the algorithm will associate each packet to an available path for a certain amount of time.

Initially, L_t contains all packets carrying the first test vector of each core. When a packet carrying a test vector is scheduled, the corresponding response packet is set to schedule at an specific time. When the response packet is scheduled, the next vector packet is included into the ready-list L_t, and this processes continues until all packets of all cores are scheduled.

The first adaptation required to the list-scheduling algorithm is the heuristic used to choose which packet will be scheduled at time t, and which packets will be deferred to be scheduled later. The heuristic used in the proposed algorithm is based on the probable test time of each core, as explained below.

Let C be the total number of cores to be tested. Each core $i, 1 \leq i \leq C$, is assigned an integer value $Cost_i$ expressing its test cost, as defined in Equation 5.6, for $Packets_i$ the number of packets of core i and $PacketSize_i$ the number of flits of each packet of core i.

$$Cost_i = Packets_i * PacketSize_i \tag{5.6}$$

Based on the values defined in Equation 5.6, the cores can be sorted in decreasing order of test cost, and the packet in L_t belonging to the core with largest test time will have priority to be scheduled. On the other hand, the access paths in each direction for each core are sorted in increasing order of length (number of routers). Thus, the scheduling algorithm will try to associate the shortest path to the most time-consuming core, to minimize its test time.

The schedule is defined as a set of time slots of different sizes as shown in Figure 5.6. The size of the time slot is measured in number of clock cycles. Each slot contains a set of packets being transmitted or processed, and the end of a slot indicates either the completion of a test or the beginning of the transmission of a new packet. One packet transmission may be distributed among several slots, and slots can be modified as the schedule is being defined. Thus, for example, one slot can be broken up into two others, to include a new packet that starts in the middle of the original slot.

The structure of the schedule is initialized as a single time slot starting at cycle zero, and with no ending time, and the initial instant of time t is set to zero. In the proposed approach, the instant time t represents one time slot in the schedule rather than a single cycle. Thus, t represents a range $[t_i..t_f]$ of test cycles, and L_t contains all packets that can be delivered at an instant time $t' \geq t_i$.

The power consumption per cycle is considered by assigning this information to each time slot of the test schedule. For each slot, the total power consumption is calculated by Equation 5.7, where n is the number of packets being transmitted during this time slot and c is the number of cores being tested in this slot.

$$Total_{power} = \sum_{1 \leq j \leq n} P_{packet}(j) + \sum_{1 \leq i \leq c} P_{core}(i) \tag{5.7}$$

Notice that a slot may have more than one packet being transmitted to/from the same core i. In this case, the power consumption of the core i (test processing) is counted only once to the slot power consumption. This is because core i can actually process only one vector at a time and the power profile of the core is assumed to be the peak consumption among all vectors. The scheduling of two or more packets related to the same core in the

same time slot means, therefore, that the packets are traversing the network, but only one of them is actually being processed by the core.

The system power limit must be respected at each time slot s, that is, $Total_{power}(s) \leq P_{max}$, where P_{max} is the maximum power consumption allowed for the system. Thus, before scheduling any packet, the total power required to transmit this packet is calculated. If the addition of this value to the total power consumption of the slot does not exceed the power consumption limit P_{max} defined for the system, that packet can be scheduled. Otherwise, the packet is set to be scheduled later.

The pseudo-code for the scheduling algorithm used in this approach is shown in Figure 5.5. Constant C is the number of cores being tested, and constants I and O represent the number of input and output ports of the system used during test. For each core, there is a list of access paths for input (IP - *Input Paths*) and output (OP - *Output Paths*), sorted as explained earlier. The list of unscheduled packets (UP - *Unscheduled Packets*) contains the input and output packets of each core, sorted according to the test cost of each module. The ready list L_t contains the packets that are ready to be delivered. The time information associated to each channel in the network is represented by the array FREE which is accessed through a channel index. These last two variables are updated as the scheduling is defined.

```
1. UP = sorted list of unscheduled packets                          O(N*logN)
2. IP [1..C][1..I] = sorted list of input routes per core           O(C*I*logI)
3. OP[1..C][1..O] = sorted list of output routes per core           O(C*O*logO)
4. Lt = list of packets ready to be scheduled
5. DT[1..C] = delivery time of the next packet of each core
6. FREE[1..TotalChannels] = time information for each channel
            in the network

7. WHILE  there are unscheduled packets                             O(N)
8.   [i,j,k] = find next pair [packet,core,path] in Lt              O(C*max(I,O))
9.   IF there is no such pair
10.    t = define next time instant                                 O(C+T0*logT0)
11.  ELSE
12.    set dijk                                                     O(1)
13.    IF power limit is respected
14.      set DT(j)                                                  O(1)
15.      set FREE(k) for each segment s of the path j               O(X+Y)
16.      update schedule (DT, Lt)                                   O(1)
17.    ELSE
18.      DT(j) = find next time instant were power constraint is met  O(T0)
19.    END
20.  END
21. END
```

Figure 5.5: Pseudo-code of the adapted list-scheduling algorithm

The algorithm starts by selecting a packet p_i in L_t belonging to core c_j. This packet has a delivery time $t_d \in t$. The shortest available path k that can be used by this packet is selected, and the duration T_{packet} of this packet in the path is defined according to Equation 5.1. If there is no available path for packet p_i, another packet belonging to another core in L_t is selected and packet p_i is set to be ready as soon as its first possible path becomes available. Otherwise, the network channels of path k are set to be unavailable during the time comprised between cycles $t_d + T_{headers} + payload$ and $t_d + T_{packet}$, according to its position in the path. If packet p_i carries a vector, the corresponding response

packet is set to be ready at time $t_d + T_{packet} - 1$, as the wrapper takes one cycle to define the header of the packet. If packet p_i is a response vector, on the other hand, the latency l of the shortest input path is calculated as $T_{headers} + T_{router} * Nb_{routers}$. Then the next input vector of core c_j is set to be ready at time $t_d - l$, ensuring that the new vector will not arrive before the previous one has been processed. In both cases, the next packet of core c_j is inserted into L_t.

The algorithm keeps trying to schedule packets at time slot t, until no more packets can be scheduled, either because all system interfaces are busy, or because there are no packets ready to be delivered at this time. When this happens, the variable t is updated to the smallest delivery time of all packets in L_t. This procedure is repeated until all packets of all cores are scheduled.

5.1.4 Example

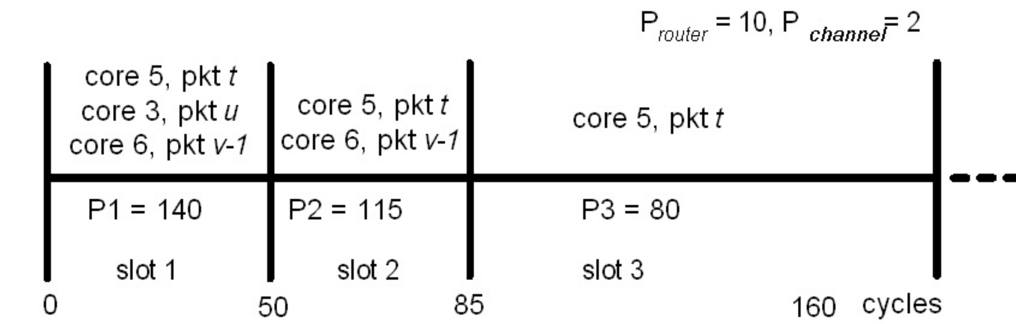
Figure 5.6 illustrates the scheduling of packet v of core 6 considering the system power constraints. For this example, let us assume that the power consumption of one SOCIN router corresponds to 10 power units, while the channel consumption corresponds to 2 power units. Moreover, let us consider the power consumption of core 6 as 50 units and the test power limit of the system as 170 units.

In Figure 5.6(a), a partial test schedule containing three time slots is shown. Three packets are currently scheduled to cores 5 (packet t), 6 (packet $v - 1$), and 3 (packet u), respectively. The transmission of packet u in the network takes 50 cycles and defines the first time slot in the schedule. Similarly, packet $v - 1$ lasts for 85 cycles and defines the second slot, while packet t defines the third slot with 160 cycles. Each slot s has the information about the total power consumed by the scheduled tasks ($P(s)$) calculated by Equation 5.7. Let us assume that variable t corresponds now to slot 1 (range $[0..50]$) and the algorithm tries to schedule as many tasks to this slot as possible. After selecting packet v for Core 6, the algorithm checks whether the power consumption of the core alone is too high for the current slot. If this is the case, the next packet of another core in the list of unscheduled packets is selected. Otherwise, an available transmission path is selected. For this example, Core 5 is the chosen system input, and the power consumption of the path from Core 5 to Core 6 is calculated by Equation 5.4, as explained earlier, and this is evaluated to 34 power units.

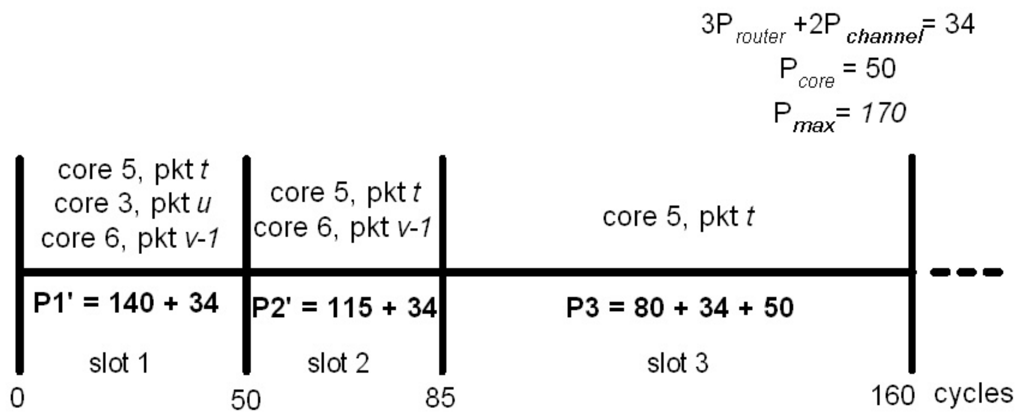
The transmission and processing of packet v is evaluated to last 85 cycles. Thus, if this communication starts at slot 1 (cycle 0), it will last until the end of slot 2. Therefore, the algorithm must check whether the power consumption limit is respected in all slots within the duration of the packet transmission. Figure 5.6(b) shows the possible power increase for each slot if packet v is scheduled to slots 1 and 2.

As Core 6 has another packet already scheduled in a slot, the contribution of the new packet to the slot power consumption corresponds only to the network consumption (routers and channels). If the core was not active in a slot, then the core consumption P_{core} would be added to Equation 5.4.

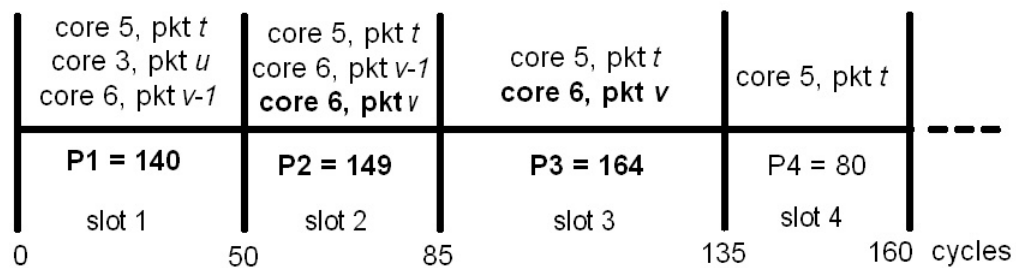
From Figure 5.6(b) one can observe that the packet can not be scheduled at slot 1, because of the power limit of 170 units. Recalculating the duration of packet v starting from slot 2, the algorithm checks slot 3 in addition to the first two slots, concluding that slots 2 and 3 can accommodate the packet. Thus, the packet is scheduled to slots 2 and 3, and the schedule is updated accordingly, as shown in Figure 5.6(c). Since the packet transmission lasts only 85 cycles, and slot 2 already offers 35 cycles, slot 3 is broken down into two new slots 3 and 4, where slot 3 now offers the 50 cycles still necessary for



(a) current schedule



(b) schedule inspection



(c) updated schedule

Figure 5.6: Scheduling process considering power constraints

the packet.

Notice that the cores that are not being tested can be shut off in order to reduce the system power consumption, since only the network structures (routers and communication channels) are active during the transmission and unpacking of the test packets. Moreover, the method does not depend on a specific test frequency. Then, the test frequency can be defined according to the system and tester constraints.

Once the schedule is defined, it is used by the external tester to assemble the test packets containing the test vectors. Each test vector carries the information required by the wrappers to assemble the test response packets, such as the address of the output interface and the delivery time information.

5.2 Complexity Analysis

Figure 5.5 also presents the complexity of each step of the algorithm for networks with 2-D topologies (grid and torus). The complexity is defined based on the following parameters: the number of scheduled packets (N), the number of tested cores (C), the number of input and output ports (I and O, respectively), the dimensions of the network (X and Y), and the total test time T_0 measured in clock cycles.

Steps 1, 2, and 3 of the algorithm require the sorting of the lists elements, which can be performed by an algorithm of logarithmic complexity with the number of elements. Steps 4, 5, and 6 define the data structure. The cost of the loop defined at step 7 is the main cost of the algorithm and is dependent on the number of packets to be scheduled. Step 8 is the heuristic that selects the next packet to be scheduled. It must check every packet in L_t and associate a path to the selected packet. The number of searches in L_t depends on the number of cores, and the search for a path depends on the number of paths defined for each core, which is a function of the number of interfaces with the tester. The definition of the next time range to be considered by the scheduling depends of the delivery times of the packets in L_t, that is, it is a function of the number of cores. However, since one packet can be set to be scheduled before the current time t, a binary search in the partial schedule is required. This search depends on the number of slots of the partial schedule. In the worst case, there are at most T_0 slots in the schedule, if all slots have a duration of one cycle. The cost of Step 15 depends on the size of the paths defined within the network, which is a function of the network dimensions. Finally, if power consumption is considered, it may be necessary to define another delivery time for the packet, and this is done by sequentially searching the scheduling for a slot with reduced power consumption. Thus, the complexity function f of the proposed algorithm can be expressed by Equation 5.8.

$$\begin{aligned} K_1 &= \max((X+Y), T_0) \\ K_2 &= \max((C + T_0 * logT_0), K_1) \\ K_3 &= C * I * logI + C * O * logO \\ K_4 &= \max(I, O) \\ f &= O\left[N * logN + K_3 + N(C * K_4 + K_2)\right] \end{aligned} \tag{5.8}$$

Since variables I, O, X, and Y are much smaller than N, and are kept constant if the number of test packets increases for the same system, the complexity reduces to $O[NlogN + N(C + T_0 * logT_0)]$

5.3 Experimental Results

The proposed scheduling algorithm was implemented in C++ and called NoCBaTe (**NoC**-**Ba**sed **Te**sting). The tool receives the system description (test requirements of the cores, power limit), the NoC description (topology, latency of the router, number and location of the interfaces with the tester), and the placement of the cores in the network. With this information, the test packets as well as the access paths to/from each core are defined, and the test schedule is generated. The current version of the NoCbaTe tool can be found in the CD-ROM attached to this manuscript, and consider two network topologies for the SOCIN network: grid and torus. However, different versions of routers (with distinct latencies or power consumption, for example) can be easily added to the algorithm. Moreover, it is important to notice that the proposed method is neither restricted to the SOCIN NoC, nor to the 2-D topologies used in this work. Any network using deterministic routing can be reused for test using this method, since all paths can be statically defined for those platforms. The inclusion of another topology in the tool is straightforward. One must only provide the routing algorithm and the arbitration policy, which will be used to define the access paths and to manage the network conflicts, respectively.

The experimental results for the ITC'02 benchmarks are presented in this section. Although the modeling of the benchmarks to the NoC-based test planning approach is considerably simpler than to the ReBaTe tool, some assumptions about the systems are still required. For example, the placement of the cores in the network is usually defined by the communication requirements of the application, which is not known for the available benchmarks. Similarly, the number of input and output ports is also unknown, since only the number of pins is informed in the benchmark description.

The same system assumptions made in Chapter 4 are reused in the experiments with the NoC. Thus, the random functional connections previously defined are interpreted as communication requirements, and, based on such requirements, a placement algorithm associates each core to a SOCIN router in the grid or torus topology. The placement algorithm for the NoC is very similar to the placement algorithm for the core-to-core connection model. It also uses a simulated annealing approach, but aims at finding the solution that minimizes the total cost of communication in the system, rather than the cost in terms of area and routing. In the experiments, the same placement used for the torus topology is reused when the grid topology is considered, by simply disconnecting the routers at the network boundaries.

The number of interfaces between the system and the tester is defined according to the system information. For the benchmarks that have a defined interface, the number of input, output, and bidirectional pins is divided into groups of 32 bits, which is the maximum channel bitwidth considered for the network. For the benchmarks that do not present this information, the interfaces defined in the random functional association defined in Chapter 4 are used in such a way that only the interfaces with 32 bits or more are considered. Again, this procedure is a simple approximation and may not represent all real cases. Actually, this may preclude the full exploration of the network parallelization. For example, if the real system has one port with less 32 bits, it will not be reused during test. However, it could be used for the network with channels of 16 bits. Moreover, even the ports with less than 16 bits can be reused if one can modify the wrapper of that port to re-define the packet in such a way that the whole communication channel is used. Since this feature is not implemented in the scheduling algorithm, the worst case is assumed for all systems.

Table 5.2: Test results for benchmark d695

Topology	Channel width (bits)	Power Limit	Test Time (cycles)
grid	32	-	17,334
		80%	16,921
		50%	17,037
		30%	27,849
	16	-	27,763
		80%	27,400
		50%	28,558
		30%	47,555
torus	32	-	16,944
		80%	16,944
		50%	16,944
		30%	17,282
	16	-	27,056
		80%	27,056
		50%	27,056
		30%	27,215

1,762 packets scheduled
3 inputs, 3 outputs

5.3.1 Benchmarks d695, g1023, f2126, q12710, and t512505

For system d695, the placement shown in Figure 5.3 is used in the experiments. The test requirements for the NoC-based testing of system d695 were presented in Table 5.1. The number of pins considered in the system interface is given in Table 4.2 and resulted in three input and three output ports, as shown in Figure 5.3.

Table 5.2 presents the resulting test costs for the benchmark d695 considering both the grid and the torus topologies. The first column indicates the network topology, the second column indicates the channel bitwidth, and the third column indicates the system power limit. The resulting testing time is presented in the fourth column. The power consumption limit is defined as explained in Chapter 4, that is, as a percentage of the sum of the power consumption of all cores.

The results of Table 5.2 show that the test time depends on the network topology and on the width of the communication channels. The network topology defines the number and length of the paths from/to the tester to/from the cores. The single difference between the grid and the torus topologies is the connection between the peripheral cores that exists in the second one. This connection provides shorter paths within the network, thus reducing the test time. The relation between the channel width and the test time is sub-linear. This happens because, for some cores, as shown in Table 5.1, the number of test pins is such that the number of flits is the same for both channel widths. Moreover, the pipeline provided by the network structure reduces the impact of the increase of the number of flits per packet.

Figure 5.7 shows the placement of benchmark g1023 in a 3x5 network, as well as the input and output ports reused during test. The test requirements of the system are summarized in Table 5.3 and the test solutions for this system are presented in Table 5.4.

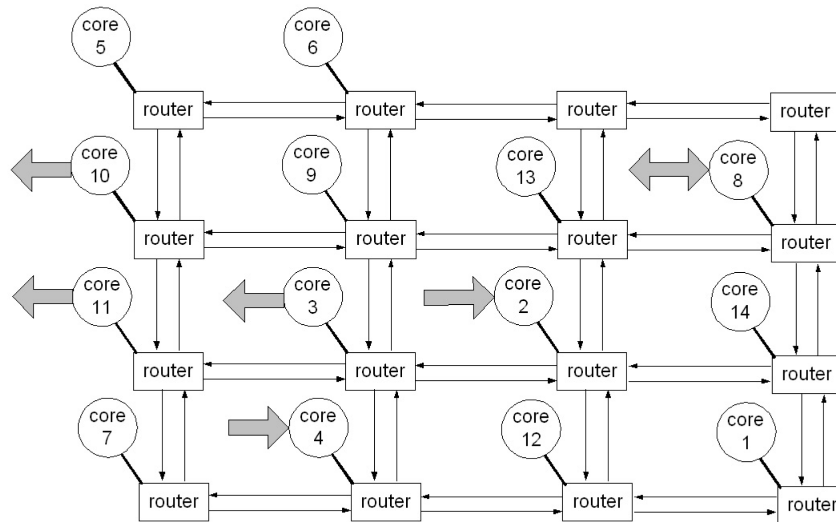


Figure 5.7: System g1023 implemented in a 4x4 NoC

Table 5.3: Test packets for system g1023

Core	# of packets	Flits/Pack. 32-bit	Flits/Pack. 16-bit
1	268	43	86
2	148	84	84
3	114	53	53
4	536	54	54
5	102	32	32
6	72	47	47
7	68	47	47
8	62	52	52
9	136	64	64
10	58	13	26
11	30	9	18
12	32	13	13
13	1,024	2	4
14	2,048	5	9

Table 5.4: Test results for benchmark g1023

Topology	Channel width (bits)	Power Limit	Test Time (cycles)
grid	32	-	41,764
		80%	41,764
		50%	41,764
		30%	42,467
	16	-	51,755
		80%	51,755
		50%	51,755
		30%	52,966
torus	32	-	27,607
		80%	27,607
		50%	27,607
		30%	27,607
	16	-	35,209
		80%	35,209
		50%	35,209
		30%	35,209

4,698 packets scheduled
2 inputs, 2 outputs

In this example, one can observe that the torus topology is less susceptible to the power limit reduction, as the test time is the same for all power configurations when this topology is used. This is probably caused by the length of the paths in this topology. As shortest routes are available, the power consumption of the network is reduced, thus keeping the original test parallelization.

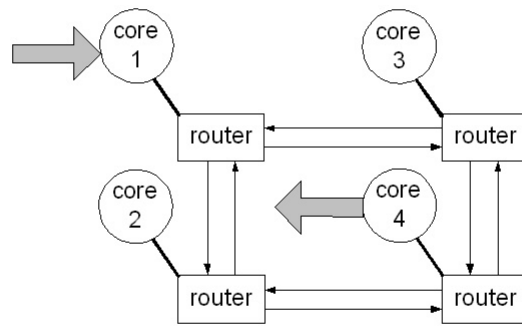


Figure 5.8: System f2126 implemented in a 2x2 NoC

Table 5.5: Test packets for system f2126

Core	# of packets	Flits/Pack. 32-bit	Flits/Pack. 16-bit
1	668	1,000	1,000
2	844	319	638
3	206	452	452
4	206	452	452

System f2126 has a trivial placement in the NoC, as shown in Figure 5.8. The test requirements of the system are presented in Table 5.5, and the test results are presented in Table 5.6.

For this system, a power constraint of 50% already poses some difficulties for the test scheduling algorithm. The power consumption of module 1 is higher than the defined power limit, and no packet of this module can be scheduled under this power constraint. A power limit of 62% is the minimum limit for which all packets of this system can be scheduled. The resulting test time for this power limit for a 32-bit grid topology is, for example, 569,595 cycles.

System q12710 has also a trivial placement in a 2x2 network. Figure 5.9 shows the placement for this benchmark while Tables 5.7 and 5.8 present the test requirements and results, respectively, for this system.

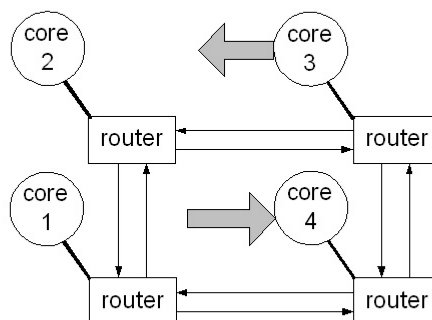


Figure 5.9: System q12710 implemented in a 2x2 NoC

Table 5.6: Test results for benchmark f2126

Topology	**Channel width (bits)**	**Power Limit**	**Test Time (cycles)**
grid	32	-	565,261
		80%	565,261
		50%	–[1]
		30%	–[1]
	16	-	699,560
		80%	700,202
		50%	–[1]
		30%	–[1]
torus	32	-	565,261
		80%	565,261
		50%	–[1]
		30%	–[1]
	16	-	699,560
		80%	699,560
		50%	–[1]
		30%	–[1]

1,924 packets scheduled
1 input, 1 output
[1] 668 unscheduled packets

Table 5.7: Test packets for system q12710

Core	**# of packets**	**Flits/Pack. 32-bit**	**Flits/Pack. 16-bit**
1	1,704	971	971
2	2,628	1,689	1,689
3	2,446	1,297	1,297
4	2,446	1,297	1,297

Table 5.8: Test results for benchmark q12710

Topology	Channel width (bits)	Power Limit	Test Time (cycles)
grid	32	-	6,230,935
		80%	6,230,935
		50%	6,231,347
		30%	–[1]
	16	-	6,230,935
		80%	6,230,935
		50%	6,231,347
		30%	–[1]
torus	32	-	6,230,935
		80%	6,230,935
		50%	6,230,935
		30%	–[1]
	16	-	6,230,935
		80%	6,230,935
		50%	6,230,935
		30%	–[1]

9,224 packets scheduled
1 input, 1 output
[1] 2628 unscheduled packets

For benchmark q12710, the power limit of 30% is higher than the power consumption of core 2 during test, preventing the scheduling of its packets. A minimum power limit of 45% is required to allow the scheduling of all packets of this benchmark. The test time for a 32-bit grid topology for the power limit of 45% is 6,231,347 cycles.

Benchmark t512505 is implemented in a 5x7 network, as shown in Figure 5.10. As the number of input and output pins is smaller than 16, only the bidirectional interface pins are reused during test, which results in four bidirectional ports. Table 5.9 presents the test requirements for the NoC-based testing, and Table 5.10 presents the resulting test times for a number of system configurations.

Table 5.9: Test packets for system t512505

Core	# of packets	Flits/Pack. 32-bit	Flits/Pack. 16-bit
1	314	389	389
2	660	104	104
3	308	904	904
4	816	740	740
5	6	10	10
6	254	154	154
7	1,216	514	514
8	2,050	1,473	1,473
9	390	530	530
10	1,576	1,264	1,264
11	376	530	530
12	84	53	53
13	136	94	94
14	556	1,225	1,225
15	302	386	386
16	740	154	154
17	160	131	131
18	306	73	73
19	158	68	68
20	154	68	68
21	484	540	540
22	466	540	540
23	1,064	1,372	1,372
24	858	1,669	1,669
25	296	190	190
26	26	4	8
27	20	2	4
28	6	1	1
29	134	5	10
30	302	302	302
31	6,740	1,550	3,100

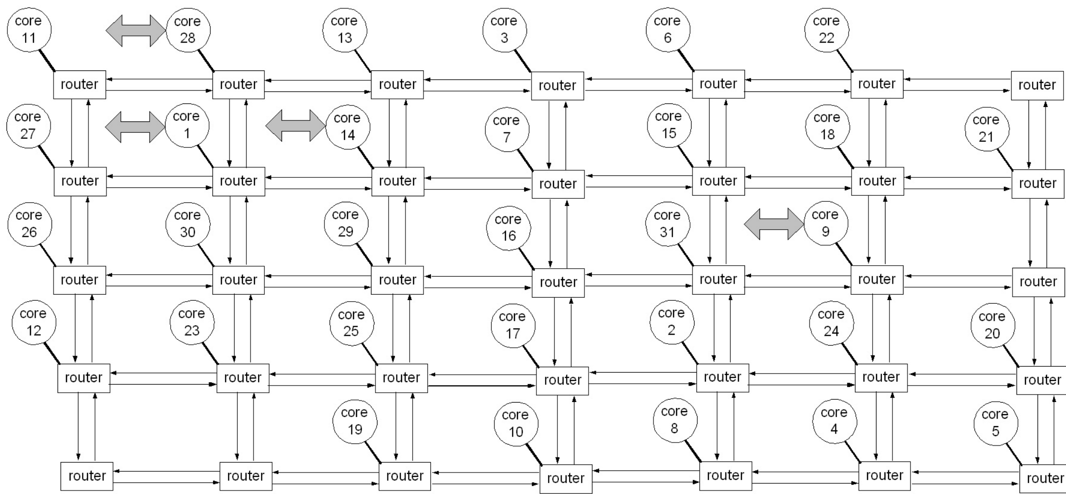


Figure 5.10: System t512505 implemented in a 5x7 NoC

Table 5.10: Test results for benchmark t512505

Topology	Channel width (bits)	Power Limit	Test Time (cycles)
grid	32	-	5,649,467
		80%	5,649,467
		50%	5,675,460
		30%	7,625,872
	16	-	10,625,082
		80%	10,625,082
		50%	10,672,428
		30%	12,852,061
torus	32	-	5,537,675
		80%	5,537,675
		50%	5,537,675
		30%	7,022,727
	16	-	10,611,103
		80%	10,611,103
		50%	10,611,103
		30%	11,675,997

20,958 packets scheduled
1 input, 1 output, 3 bidirectional ports

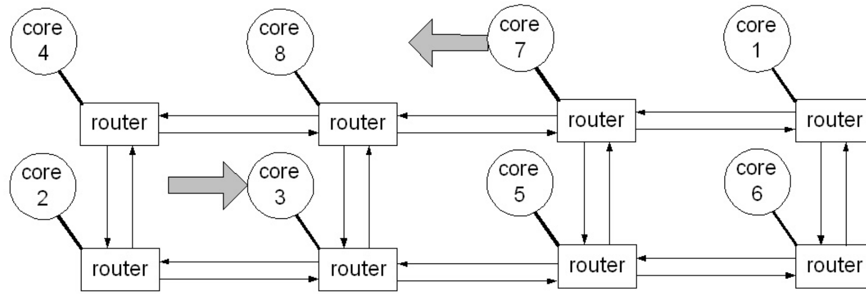


Figure 5.11: System h953 implemented in a 2x4 NoC

Table 5.11: Test packets for system h953

Core	# of packets	Flits/Pack. 32-bit	Flits/Pack. 16-bit
1	682	348	348
2	18	327	327
3	78	32	32
4	98	21	21
5	220	121	121
6	364	185	185
7	130	3	5
8	610	189	189

5.3.2 Benchmark h953

Figure 5.11 shows the placement of system h953 and Table 5.11 presents the test requirements of this system. Table 5.12 shows the test solutions found for this benchmark, considering one input and one output interfaces with the external tester.

Module 2 in system h953 has a power consumption that exceeds the power limit of 30%. The limit of 50% is the minimum power consumption limit that assures the scheduling of all packets.

5.3.3 Benchmarks u226 and d281

Benchmarks u226 and d281 contain BISTed cores. For such modules, there are only two messages that must transit between the external tester and the core: a test enable indication and the resulting signature.

A BISTed core is modeled in the NoC-based test planning approach as any other core in the system. The only difference is that such a module has a very reduced number of packets, and the packets have a payload of only one flit. In addition, the test time of the BISTed module replaces Equation 5.6, but the same priority rules of the other cores are followed. At last, the single response packet of this block is ready to be sent only after the BIST test is complete. All other steps of the algorithm are performed with no change.

Again, to model a core with more than one test (BISTed or external), the same procedure defined in Section 4.7.3 can be used. One can define a virtual core with the same location of the actual module, but with different test packets.

The placement of benchmark u226 in a 3x3 network is shown in Figure 5.12. Ta-

Table 5.12: Test results for benchmark h953

Topology	Channel width (bits)	Power Limit	Test Time (cycles)
grid	32	-	231,278
		80%	231,278
		50%	231,556
		30%	– [1]
	16	-	231,407
		80%	231,407
		50%	231,685
		30%	– [1]
torus	32	-	231,278
		80%	231,278
		50%	231,278
		30%	– [1]
	16	-	231,407
		80%	231,407
		50%	231,407
		30%	– [1]

2,200 packets scheduled
1 input, 1 output
[1] 18 unscheduled packets

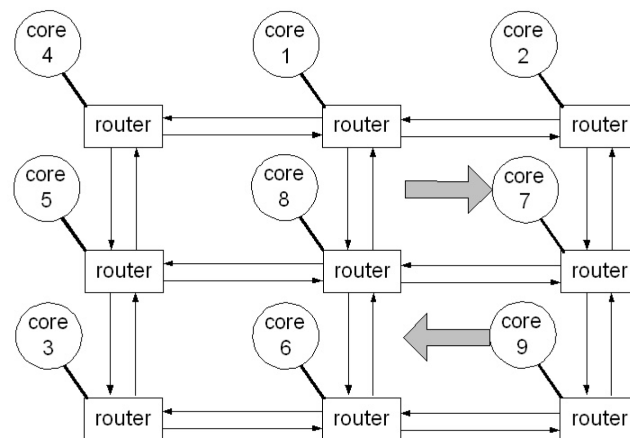


Figure 5.12: System u226 implemented in a 3x3 NoC

Table 5.13: Test packets for system u226

Core	# of packets	Flits/Pack. 32-bit	Flits/Pack. 16-bit
1	2	1	1
2	2	1	1
3	2	1	1
4	5,336	1	2
5	5,336	1	2
6	5,336	1	2
7	152	52	104
8	2	1	1
9	30	1	2

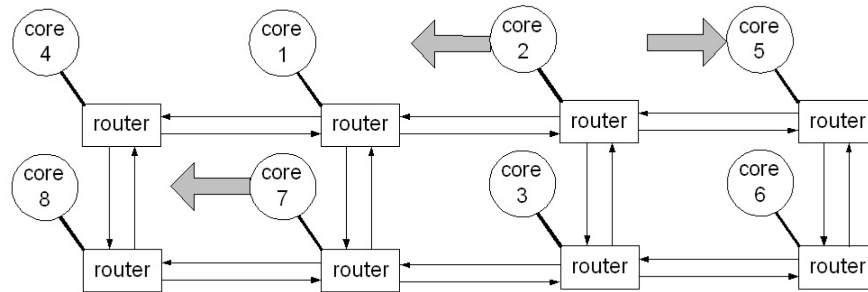


Figure 5.13: System d281 implemented in a 2x4 NoC

ble 5.13 presents the test requirements of this system, and Table 5.14 presents the test solutions considering one input and one output ports.

System u226 also shows an incomplete test scheduling due to the power limit of 30%. All unscheduled packets belong to Module 7. However, in this case, the power consumption of the core is not higher than the system limit, but it is very close to it. Therefore, the consumption of any path used to access this core added to the core consumption will result in a power consumption higher than the established limit. A complete test schedule can be defined for this system for a power limit as low as 35%.

The implementation of the benchmark d281 in a 2x4 network is shown in Figure 5.13.

The number of test packets per core for each test, as well as the other test requirements of this system are shown in Table 5.15. In the table, the virtual cores that implement the second test of some modules are indicated by the *a* suffix. For example, module 1a is the BIST testing of module 1, that already contains some test packets implementing the external testing.

Table 5.16 presents the resulting test costs for the benchmark d281.

Table 5.14: Test results for benchmark u226

Topology	Channel width (bits)	Power Limit	Test Time (cycles)
grid	32	-	1,392,151
		80%	1,419,467
		50%	2,782,770
		30%	–[1]
	16	-	1,404,116
		80%	1,426,351
		50%	2,786,724
		30%	–[1]
torus	32	-	1,392,148
		80%	1,394,079
		50%	1,410,127
		30%	–[1]
	16	-	1,404,112
		80%	1,404,112
		50%	1,414,873
		30%	–[1]

16,186 packets scheduled
1 input, 1 output
[1] 152 unscheduled packets

Table 5.15: Test packets for system d281

Core	# of packets	Flits/Pack. 32-bit	Flits/Pack. 16-bit
1	52	2	4
1a	2	1	1
2	316	8	15
2a	2	1	1
3	192	7	13
3a	2	1	1
4	180	8	8
4a	2	1	1
5	236	32	32
5a	2	1	1
6	160	9	9
6a	2	1	1
7	2	1	1
8	116	9	9
8a	2	1	1

Table 5.16: Test results for benchmark d281

Topology	**Channel width (bits)**	**Power Limit**	**Test Time (cycles)**
grid	32	-	9,024
		80%	9,024
		50%	9,024
		30%	9,028
	16	-	10,769
		80%	10,769
		50%	10,769
		30%	10,772
torus	32	-	9,029
		80%	9,029
		50%	9,029
		30%	9,029
	16	-	10,764
		80%	10,764
		50%	10,764
		30%	10,764

1,268 packets scheduled
1 input, 2 output ports

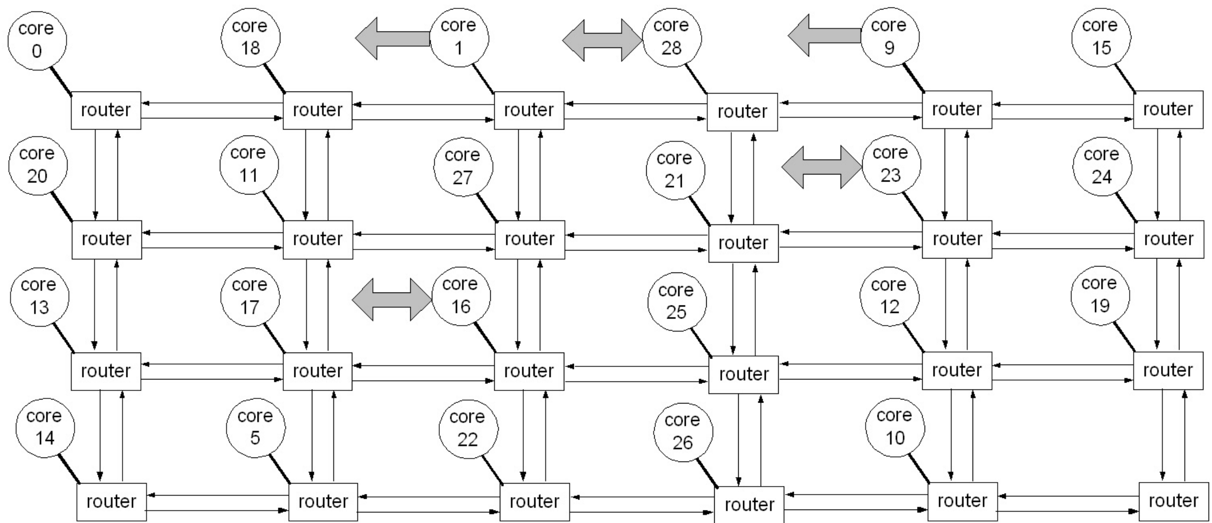


Figure 5.14: System p22810 implemented in a 4x6 NoC

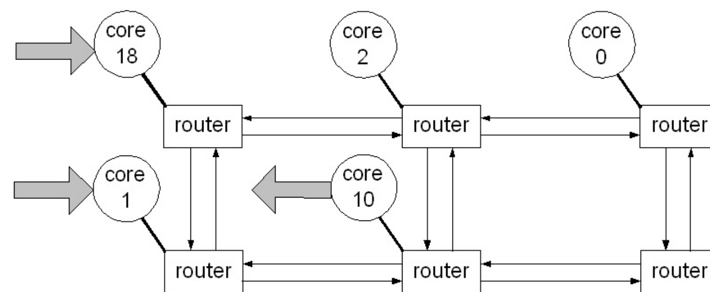


Figure 5.15: System p34392 implemented in a 2x3 NoC

5.3.4 Benchmarks p22810, p34392, p93791, and a586710

Modules embedded into super-cores are also modeled as virtual cores located in the same router of the corresponding super-core, assuming that one can access the lower-level core interface through the wrapper of the higher-level block.

System p22810 implemented in a 4x6 NoC is shown in Figure 5.14, where only the super cores are associated to the network routers. Table 5.17 presents the test requirements of the cores in this system. The second column in this table indicates which level of the system hierarchy each core is located in. System-level testing is modeled as another core, indicated as Core 0 in Figure 5.14, and it contains two tests implemented by Module 0.

Table 5.18 presents the resulting test costs for the benchmark p22810. As the system has only 10 input pins, only the bidirectional and output pins were reused during test, resulting in 3 bidirectional and 2 output ports.

The placement defined for the benchmark p34392 in a 2x3 network is shown in Figure 5.15 while Tables 5.19 and 5.20 present the test requirements of the cores and the resulting test solution found for this system, respectively.

Figure 5.16 shows the placement of system p93791 in a 3x5 NoC. Table 5.21 summarizes the test requirements of the embedded modules, and Table 5.22 presents the test solutions found with the proposed test planning tool.

Table 5.17: Test packets for system p22810

Core	Level	# of packets	Flits/Pack. 32-bit	Flits/Pack. 16-bit
0	0	198	6	2
1	1	1,570	130	130
2	2	24,648	3	6
3	2	6,216	3	6
4	2	444	2	4
5	1	404	214	428
6	2	1,424	3	5
7	2	5,264	3	5
8	2	5,216	2	3
9	1	350	123	246
10	1	76	99	99
11	1	188	88	88
12	1	186	82	82
13	1	2	104	104
14	1	216	78	78
15	1	74	80	80
16	1	16	109	109
17	1	50	89	89
18	1	1,288	68	68
19	1	116	43	43
20	1	248	77	77
21	1	930	186	186
22	1	118	77	77
23	1	80	115	115
24	1	54	103	103
25	1	430	181	362
26	1	362	800	1,600
27	1	4	34	34
28	1	52	100	100

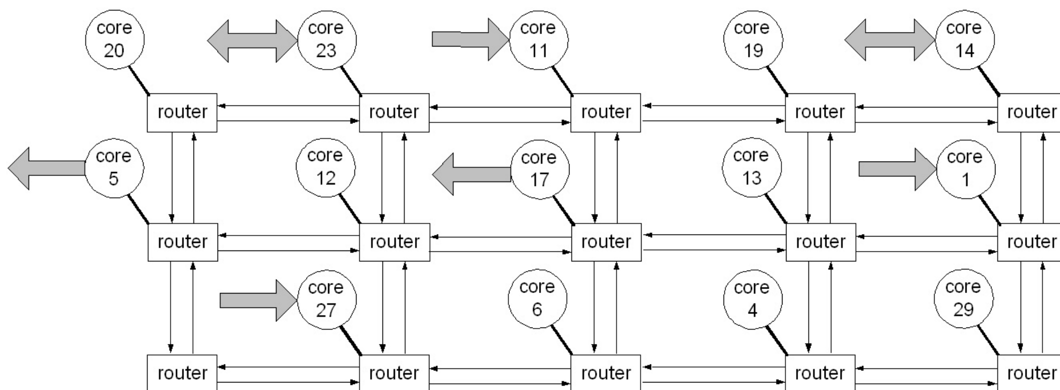


Figure 5.16: System p93791 implemented in a 3x5 NoC

Table 5.18: Test results for benchmark p22810

Topology	Channel width (bits)	Power Limit	Test Time (cycles)
grid	32	-	371,814
		80%	371,814
		50%	371,814
		30%	375,928
	16	-	499,101
		80%	499,101
		50%	499,101
		30%	509,908
torus	32	-	364,507
		80%	364,507
		50%	364,507
		30%	364,507
	16	-	490,780
		30%	490,780
		50%	490,780
		30%	490,780

50,026 packets scheduled
1 input, 2 outputs, and 2 bidirectional ports

Table 5.19: Test packets for system p34392

Core	Level	# of packets	Flits/Pack. 32-bit	Flits/Pack. 16-bit
0	0	54	5	10
1	1	420	806	806
2	1	1,028	570	1,140
3	2	6,216	2	3
4	2	12,360	2	3
5	2	24,672	2	4
6	2	3,930	1	1
7	2	1,024	1	1
8	2	19,860	2	3
9	2	456	2	3
10	1	908	519	519
11	2	18,570	1	2
12	2	346	1	1
13	2	5,120	1	1
14	2	864	1	1
15	2	8,880	1	2
16	2	256	1	1
17	2	1,572	1	1
18	1	1,490	729	729
19	2	24,672	2	4

Table 5.20: Test results for benchmark p34392

Topology	Channel width (bits)	Power Limit	Test Time (cycles)
grid	32	-	764,570
		80%	764,570
		50%	834,060
		30%	1,117,038
	16	-	977,845
		80%	977,845
		50%	1,083,348
		30%	1,375,612
torus	32	-	764,570
		80%	764,570
		50%	764,570
		30%	755,872
	16	-	977,845
		80%	977,845
		50%	977,845
		30%	996,834

132,644 packets scheduled
2 inputs, 2 outputs

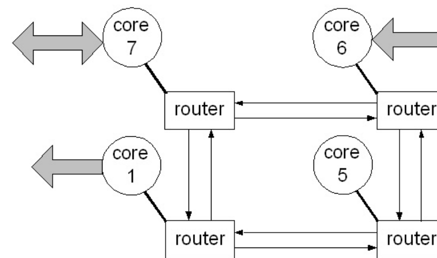


Figure 5.17: System a586710 implemented in a 2x2 NoC

System a586710 placement in a 2x2 network is shown in Figure 5.3.4. Table 5.23 summarizes the test requirements of this system.

Benchmark a586710 has 4,283,788 packets to be scheduled and the scheduling algorithm could not complete in a 1.3 GHz AMD processor with 512Mbytes of RAM memory.

We can observe in the previous results that the higher the number of packets to be scheduled, the better is the gain of the torus topology over the grid one. Indeed, as the average length of the access paths within the torus network is smaller than for in the grid topology, the average test time of each core is also smaller for the former. Furthermore, the torus topology is less susceptible to the power constraints, also because of the average length of the internal paths. On the other hand, the assumption that the network channels have the same length (when calculating the power consumption of the network during test) may be less accurate to the torus topology than to the grid one, because of the con-

Table 5.21: Test packets for system p93791

Core	Level	# of packets	Flits/Pack. 32-bit	Flits/Pack. 16-bit
0	0	0	0	0
1	1	88	336	672
2	2	1,296	2	3
3	2	354	1	2
4	1	22	10	20
5	1	12,254	6	11
6	1	436	1,042	2,084
7	2	354	1	2
8	2	384	2	3
9	2	2,328	9	17
10	1	374	82	82
11	1	782	186	372
12	1	388	438	876
13	1	388	438	876
14	1	516	300	600
15	2	576	2	3
16	2	792	5	9
17	1	420	200	400
18	2	84	3	5
19	1	832	362	724
20	1	468	350	700
21	2	84	3	5
22	2	84	2	3
23	1	1,832	136	272
24	2	6,144	1	2
25	2	5,376	1	2
26	2	192	2	3
27	1	344	378	756
28	2	792	4	7
29	1	384	4	7
30	2	384	2	3
31	2	408	5	10
32	2	6,168	9	17

Table 5.22: Test results for benchmark p93791

Topology	Channel width (bits)	Power Limit	Test Time (cycles)
grid	32	-	449,422
		80%	449,422
		50%	450,439
		30%	493,818
	16	-	830,107
		80%	830,107
		50%	832,949
		30%	860,201
torus	32	-	412,829
		80%	412,829
		50%	412,829
		30%	412,829
	16	-	778,113
		80%	778,113
		50%	778,113
		30%	778,113

46,058 packets scheduled
3 inputs, 2 outputs, and 2 bidirectional ports

Table 5.23: Test packets for system a586710

Core	Level	# of packets	Flits/Pack. 32-bit	Flits/Pack. 16-bit
0	0	0	0	0
1	1	5,890	2,155	2,155
2	2	2	11	21
3	2	2	1	1
4	2	362,280	1	1
5	1	5,890	2,626	2,626
6	1	80,862	2	3
7	1	3,828,866	8	15

Table 5.24: d695 test time for different placements in the network

Exp.	Topology	Channel width (bits)	Test Time (cycles)
d695_place1	grid	32	29,945
		16	44,184
d695_place1	torus	32	24,830
		16	40,947
d695_place2	grid	32	24,031
		16	40,130
d695_place2	torus	32	25,238
		16	39,959

1,762 packets scheduled

nections among the boundary routers present in the torus design. A more accurate model is necessary to confirm that the torus topology actually consumes less power during test.

5.4 System Configurations and Resulting Test Time

Some additional experimental results are presented in this section. Such experiments aim at evaluating the impact of a number of system configurations on the system test time. The following situations will be considered: the placement of the cores in the network, the number of interfaces with the tester, and the test under power constraints.

5.4.1 Placement of the Cores in the Network

Different placements mean different distances (number of routers) between each core under test and the external tester. Table 5.24 presents the variation in the test time of system d695 when different placements of the cores in the network are considered. Experiment *d695_place1* refers to the system shown in Figure 5.18(a), and *d695_place2* refers to the configuration shown in Figure 5.18(b). The same interfaces (at cores 3, 6, 4, and 7) with the tester are used in both cases. Again, no power constraint is assumed.

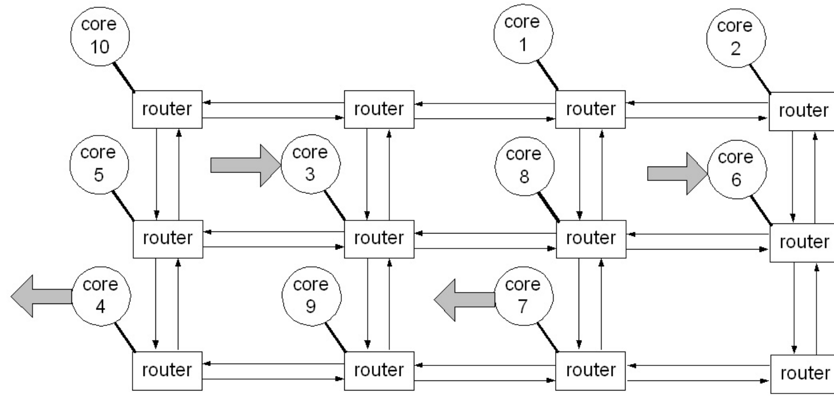
One can observe that in general, the placement has a small influence on the system test time. In most cases, the difference in the test time is smaller than 10%. The only exception is for the 32-bit grid topology, with a difference of 20% between the two placements.

5.4.2 Number of Interfaces with the Tester

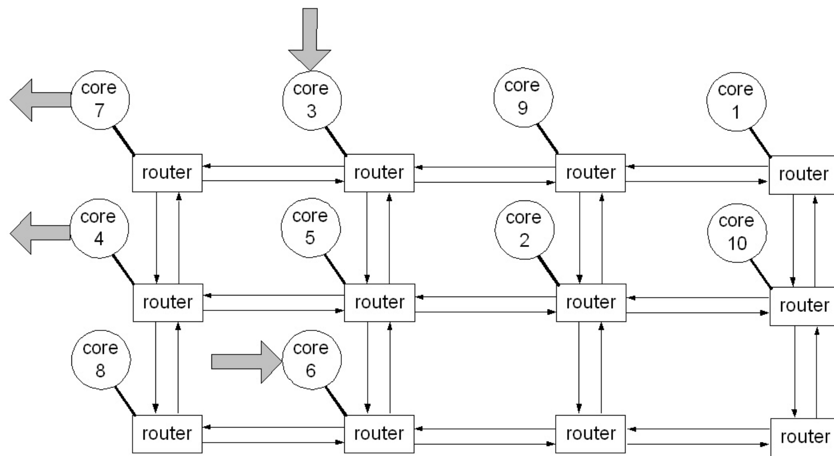
Figure 5.19 shows the variation of the test time for the system of Figure 5.3, as the number of interfaces with the tester increases. For this experiment, only uni-directional ports were considered.

These results show that the test time is very dependent on the number of input and output interfaces with the tester, since this number defines the amount of initial parallel paths that can be used during test. As the number of interfaces increases, test time decreases. This information can be further explored by the system integrator to reduce the system test time. For example, extra interfaces can be created, in addition to the reuse of the functional ports.

Table 5.25 presents the system test time when the interfaces of system t512505 are modified. This benchmark has 15 input, 13 output, and 132 bidirectional pins. In Ta-

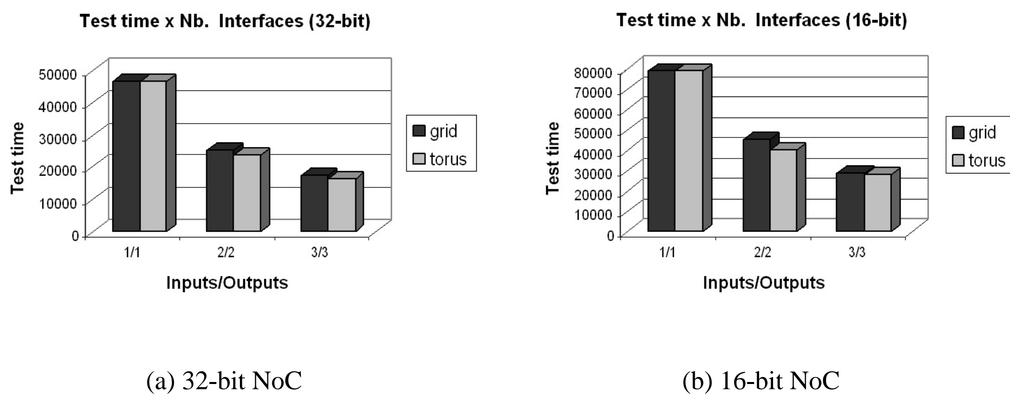


(a) d695_place1



(b) d695_place2

Figure 5.18: Different placements for system d695 in a 3x4 network



(a) 32-bit NoC

(b) 16-bit NoC

Figure 5.19: d695 test time variation with the number of interfaces with the tester

ble 5.25, the first column indicates the configuration of the test interfaces in terms of number of input, output, and bidirectional ports of 32 bits distributed among the functional pins defined in the SoC description.

Table 5.25: Test time of t512505 for different interface configurations

Configuration in/out/bidir.	Topology	Channel width (bits)	Test Time (cycles)
3/2/0	grid	32	6,521,410
		16	11,185,613
	torus	32	5,569,788
		16	10,486,612
1/1/3	grid	32	5,649,467
		16	10,625,082
	torus	32	5,537,675
		16	10,611,103
0/0/4	grid	32	6,175,913
		16	11,085,464
	torus	32	6,198,855
		16	11,001,571
2/2/0	grid	32	5,842,893
		16	11,405,558
	torus	32	6,582,433
		16	10,987,061

20,958 packets, 5x7 topology

Analyzing the first two configurations of Table 5.25, one can notice a reduction in the test time when more ports are considered. Indeed, although the total number of ports is 5 for both cases, the actual number of physical connections with the tester in each direction is increased in the second configuration, thus increasing the number of possible access paths to each module. On the other hand, the use of four bidirectional ports is worse than the use of one unidirectional and three bidirectional interfaces, since the input packets will initially occupy all ports and generate a bottleneck in the output paths. However, these results confirm that the number of interfaces actually defines the system test time. One can observe, for instance, that it is usually better to have a higher number of interfaces in total than to have less unidirectional ports, as the last configuration exemplifies.

5.4.3 Network Power Profile

Considering the high level of test parallelization achieved by the network reuse, the system power constraint is another variable that may affect the system test time. The following results show the variation of the system test time according to two parameters: the maximum test power consumption allowed for the system, and the number of system interfaces with the tester. The power limit varies from 10% to 50% of the sum of the power consumption of all cores in the system during test. The number of interfaces varies from two (one input and one output) to eight (four inputs and four outputs), and a communication channel of 32 bits is used.

The core and wrapper consumption per vector per cycle are considered as the peak consumption among all vectors.

In addition, two possibilities of power consumption for the network structures were

Table 5.26: Test times for d695: cores consumption >> routers consumption

Inputs/ Outputs	No power Limit	50%	30%	20%	10%
1/1	46,648	46,648 (0%)	46,815 (0.36%)	-[1]	-[2]
2/2	26,012	27,087 (4.13%)	28,569 (9.8%)	-[1]	-[2]
3/3	20,753	20,733 (-0.096%)	32,159 (54.9%)	-[1]	-[2]
4/4	14,785	17,623 (19.2%)	28,787 (94.7%)	-[1]	-[2]

[1] 24 unscheduled packets out of 1762
[2] 964 unscheduled packets out of 1762

Table 5.27: Test times for g1023: cores consumption >> routers consumption

Inputs/ Outputs	No power Limit	50%	30%	20%	10%
1/1	52,145	52,145 (0%)	52,296 (0.29%)	-[1]	-[2]
2/2	31,898	31,547 (-1.1%)	33,032 (3.56%)	-[1]	-[2]
3/3	22,648	24,869 (9.8%)	25,873 (14.2%)	-[1]	-[2]
4/4	18,851	19,776 (4.9%)	31,488 (67%)	-[1]	-[2]

[1] 58 unscheduled packets out of 4698
[2] 650 unscheduled packets out of 4698

considered in the experimental setup. In the first situation, the core consumption is assumed to be one order of magnitude higher than the consumption of a SOCIN router, and two orders higher than the channel consumption for the same network. This is the model used in all experiments of Section 5.3.

Tables 5.26, 5.27, and 5.28 show the variation of the system test time for this first experiment for benchmarks d695, g1023, and p22810, respectively. For the cases where all packets were scheduled, the difference from the original test time is given as a percentage.

From those results, the following conclusions can be drawn:

- power constraints can change the original order of the scheduled cores, leading the scheduling heuristic to a better solution in some particular cases. This is the case, for example, of system g1023 when 4 interfaces are used (a reduction of 1.1% in test time). As detailed in Section 5.1, the packets are firstly sorted and selected in a certain order to be scheduled. If a packet is not scheduled because of the test power limit, it is set to be scheduled later, thus changing the original order of the unscheduled cores list. For some cases, the new scheduling intensifies the pipeline in the core internal scan chains(more vectors are serially delivered), thus reducing

Table 5.28: Test times for p22810: cores consumption >> routers consumption

Inputs/ Outputs	No power Limit	50%	30%	20%	10%
1/1	549,350	549,350 (0%)	549,389 (0.0071%)	549,569 (0.04%)	-[1]
2/2	315,708	315,708 (0%)	357,239 (13.2%)	390,156 (23.6%)	-[1]
3/3	222,432	224,411 (0.9%)	222,338 (-0.042%)	349,374 (57%)	-[1]
4/4	170,999	177,330 (3.7%)	243,031 (42.1%)	331,777 (94%)	-[2]

[1] An average of 89% of the packets not scheduled
[2] 1116 unscheduled packets out of 6814

the core test time. Indeed, as the network consumption is much smaller than the core consumption, it is better to schedule another packet for the same core than schedule a packet for another core that is not yet present in the time slot;

- very tight power constraints may either increase the test time or prevent the scheduling of some test packets. In the first case, the core consumption is within the system limits, but the core and network consumption together exceed the power limit. Thus, the scheduling of some packets must be delayed to time slots with less packets. One can observe this situation, for example, when the power limit is 30% for systems d695 and g1023, and 20% for system p22810. In the second case, the power consumption of some cores is greater than the system power limit. Hence, the test packets of those cores can not be scheduled, even if they are the single packet in the time slot. It means that in this case, the consumption of the TAM is not a problem, but the consumption of the core itself is. For systems d695 and g1023, a power limit below or equal to 20% prevents the scheduling of all packets. For a larger system, such as p22810, this happens only for a power limit of 10%.

- the more interfaces with the tester, the higher the impact of the power constraints on the system test time. One can observe that the increase in the test time in the last two lines of Tables 5.26 and 5.27 is higher than for the first two lines. For system d695 and a power limit of 30%, for instance, one can notice an increase of 94.7% in the test time for eight interfaces and only 9.8% for four interfaces. In fact, the power limit may preclude the total usage of the network resources, thus increasing test time. If less interfaces are available, the parallelism is already restricted and the test time is not deeply affected;

- finally, although one can notice an increase in the test time caused by power constraints, the network still presents a very effective trade-off in terms of pins and area overhead and test time. One can observe that the increase in the system test time is very small for a power limit as low as 50%. Of course, as the power limit reduces, the system test time increases, but not in the same proportion.

The second power consumption model considered in the experiments, assumes the consumption of the cores is in the same magnitude of the routers consumption. This

may be the case, for example, of systems with a large number of small embedded cores. Tables 5.29, 5.30, and 5.31 present the test times for systems d695, g1023, and p22810, respectively, for this second situation.

Table 5.29: Test times for d695: cores consumption≈routers consumption

Inputs/ Outputs	No power Limit	50%	30%	20%	10%
1/1	46,648	47,012	74,935	-[1]	-[2]
2/2	26,012	28,746	55,139	-[1]	-[2]
3/3	20,753	27,228	47,530	-[3]	-[2]
4/4	14,785	22,802	46,952	-[3]	-[2]

[1] 379 unscheduled packets out of 1762
[2] An average of 25% of the packets not scheduled
[3] 160 unscheduled packets out of 1762

Table 5.30: Test times for g1023: cores consumption≈routers consumption

Inputs/ Outputs	No power Limit	50%	30%	20%	10%
1/1	52,145	-[1]	-[2]	-[2]	-[2]
2/2	31,898	-[1]	-[1]	-[1]	-[2]
3/3	22,648	-[1]	-[1]	-[1]	-[2]
4/4	18,851	-[1]	-[1]	-[1]	-[2]

[1] An average of 90% of the packets are not scheduled
[2] no packet scheduled

Table 5.31: Test times for p22810: cores consumption≈routers consumption

Inputs/ Outputs	No power Limit	50%	30%	20%	10%
1/1	549,350	-[1]	-[1]	-[2]	-[3]
2/2	315,708	-[1]	-[1]	-[2]	-[3]
3/3	222,432	-[1]	-[1]	-[2]	-[3]
4/4	170,999	-[1]	-[1]	-[2]	-[3]

[1] An average of 5% of the packets are not scheduled
[2] An average of 14% of the packets are not scheduled
[3] An average of 56% of the packets are not scheduled

The conclusions drawn from the first experiment (Tables 5.26 to 5.28) are still valid. However, one can observe that the test scheduling is totally compromised, as the majority of the test packets could not be scheduled for larger systems such as g1023 and p22810. In this case, the power consumption of the network structure becomes clear. Although the number of active cores per time slot in the schedule is limited by the system interfaces, the number of active network routers and channels may be considerably large,

since several packets are being transmitted at different points in the communication platform. For instance, the average number of cores per slot in the schedule with no power constraint in system g1023 with 6 interfaces is 3.4, while the average number of packets per slot being transmitted through routers and channels is 35.9 (from the 48 possible ones). When the power consumption is limited to 30%, for the same number of inputs and outputs, the traffic in the network through routers and channels reduces to 1.5 for the same example. Unlike the results of Tables 5.27 and 5.28, when the power limit is 50% in Tables 5.30 and 5.31 (systems g1023 and p22810 for this second power consumption model), the consumption of the cores is not greater than the system limit, but it is very close to it. Therefore, any network resource used to transmit a packet will exceed the system power limit. This means that only the packets of the cores that are at the system interface are scheduled, since they are the ones that require less network resources to be tested.

Table 5.32 presents the results for the system p93791 considering 3 input, 2 output, and 2 bi-directional ports in a 32-bit SOCIN network. For this number of interfaces, three possible power consumption models for the embedded cores are considered. In the first case, the consumption of all cores is one order of magnitude higher than the consumption of the network routers. In the second case, the consumption of all cores is similar to the consumption of the routers. Then, a mixed situation is considered, that is, some cores present higher power consumption, while smaller cores have a power consumption similar to the routers. A hierarchical description of the system is assumed, that is, cores embedded into other cores are not directly connected to a network router and have their test vectors transmitted through the wrapper of their associated super core.

Table 5.32: Test times for p93791: 5 inputs and 4 outputs

Power model	**No power Limit**	**50%**	**30%**	**20%**
cores>>routers	435,787	434,820	445,845	-*
cores≈routers	435,787	478.535	778,655	-*
mixed	435,787	439,965	564,030	-*

* 436 packets unscheduled out of 46058 packets

One can observe from Table 5.32 that the increase in the system test time in the first power model (*cores>>routers*) is of only 2.3% for a power limit as low as 30% of the sum of the power consumption of all cores in the system. When the consumption of all cores is reduced (*cores≈routers*), the increase in the test time is higher, 78.7%. However, it is very likely that in a real system, the power consumption of the cores has a mixed distribution, as presented in the third line of Table 5.32. For this case, the system test time is practically the same for a system with no power limit and a system with a power limit of 50%. For a power limit reduced to 30%, the increase in the system test time is of 29.4%.

6 DISCUSSION

In this chapter, the proposed approaches are compared to each other and to other test approaches that have used the ITC'02 SoC Test benchmarks as well.

6.1 Reuse-based versus NoC-based Test Planning

Table 6.1 presents some comparative results between the two proposed test planning methods. As the NoC-based technique only optimizes the system test time, this is the main parameter used for comparison. It is assumed, however, that the area overhead of the NoC-based test is of the same magnitude or smaller than the area overhead generated by the ReBaTe tool. As for the number of extra pins in the system interface, the NoC-based method presents no pin overhead, while the number of extra pins for the ReBaTe can be quite variable. Therefore, it is possible to compare the test time of the NoC-based test with the test time of the cheapest solution found by the ReBaTe tool. Hence, for comparable costs (smallest pin overhead and almost the same area overhead), one can evaluate which method presents the best test time.

For the ReBaTe tool (Chapter 4), the solutions with the smallest number of pins for the combinations of pins, time, and area optimization are considered in all cases, with or without power consumption control. For the NoCbaTe method (Chapter 5), the solutions for the 32-bit and 16-bit grid topology are used. In Table 6.1, the smallest test times among the three possibilities are presented in bold. The comparison is not presented for benchmark a586710, since there is no available results for this system using the NoCbaTe tool.

It is interesting to observe in Table 6.1 that each method generates the best test for half of the cases. Thus, from the 29 system configurations compared, 15 have the best test time generated by the NoCbaTe test planning, while the remaining 14 cases have the best test time generated by the ReBaTe tool. Furthermore, it is interesting to notice that one method can be better than the other depending on the configuration of the system, rather than on the system itself. For example, for system p34392 the NoC-based test gives the best test time for a power limit as low as 50%. For the power limit of 30%, the ReBaTe method gives the best result. However, it is important to have in mind the pin overhead generated by the ReBaTe approach. For system p34392, this overhead is of 49 pins, for a gain of only 7% in the test time. Systems d695, f2126, and d281 present the same situation.

One can argue that if the same number of extra pins generated by the ReBaTe approach is added to the system interface to be used by the NoC-based test, better test times can be achieved by the later method. This is true for some cases. For example, let us analyze the benchmark p34392. With 49 extra pins, one can add one port of 32 bits to the 32-bit

Table 6.1: Comparative results between proposed approaches

Benchmark	Power	ReBaTe tool		NoCbaTe tool	
	limit	Test Pins	Test time	32-bit	16-bit
d695	-	3	37,089	**17,334**	27,763
	80%	3	37,089	**16,921**	27,400
	50%	1	31,021	**17,037**	28,558
	30%	94	**25,147**	27,849	47,555
g1023	-	3	157,120	**41,764**	51,755
	80%	3	157,120	**41,764**	51,755
	50%	4	**39,844**	41,764	51,755
	30%	17	48,501	**42,467**	52,966
f2126	-	17	566,105	**565,261**	699,560
	80%	18	**518,858**	565,261	700,202
	62%	36	**470,693**	569,595	704,213
q12710	-	0	**4,645,404**	6,230,935	6,230,935
	80%	0	6,236,421	**6,230,935**	6,230,935
	50%	1	**3,313,904**	6,231,347	6,231,347
t512505	-	13	7,485,064	**5,649,467**	10,625,082
	80%	14	6,092,953	**5,649,467**	10,625,082
	50%	19	6,267,593	**5,675,460**	10,672,428
	30%	8	14,278,685	**7,625,872**	12,852,061
h953	-	2	**223,139**	231,278	231,407
	80%	3	**148,966**	231,278	231,407
	50%	3	**218,391**	231,556	231,685
u226	-	3	**1,363,968**	1,392,151	1,404,116
	80%	3	**1,363,968**	1,419,467	1,426,351
	50%	3	**1,460,140**	2,782,770	2,786,724
	41%	3	**1,363,968**	3,898,706	3,904,043
d281	-	2	30,784	**9,024**	10,769
	80%	2	30,784	**9,024**	10,769
	50%	6	12,320	**9,024**	10,769
	30%	26	**8,132**	9,028	10,772
p22810	-	76	**281,626**	371,814	499,101
	80%	76	**281,626**	371,814	499,101
	50%	67	385,389	**371,814**	499,101
	30%	78	**290,398**	375,928	509,908
p34392	-	2	1,737,079	**764,570**	977,845
	80%	2	1,737,079	**764,570**	977,845
	50%	2	1,802,461	**834,060**	1,083,348
	30%	49	1,368,792	**1,117,038**	1,375,612
p93791	-	77	530,667	**449,422**	830,107
	80%	130	523,045	**449,422**	830,107
	50%	100	586,738	**450,439**	832,949
	30%	11	1,615,500	**493,818**	860,201

[1]ReBaTe optimization factors: pins, test time, and area
[2]NoC topology: grid

NoC, or at most two ports of 16 bits to the 16-bit NoC. Let us assume an input port is added to the system in the 32-bit NoC, and one input and one output port are added in the 16-bit NoC. The resulting test time is, respectively, 1,117,038 and 1,375,937 cycles. Hence, for this case, the pin overhead would generate a better solution only for the 32-bit NoC. For system q12710, on the other hand, the NoC-based test time is 88% higher than the ReBaTe solution. However, the pin overhead of the ReBaTe method is only 1, which will not improve the results of the NoC-based approach.

Another conclusion from Table 6.1 is that the ReBaTe approach usually generates the best test times for smaller power limits (systems q12710, u226, p34392, for instance) whereas the NoCBaTe method is more efficient for configurations with higher power limits. Moreover, the gain of the NoC-based approach, mainly for higher power limits, is usually more significant than the gain of the ReBaTe tool, as it happens for systems d695 and d281, for example.

The execution time of the RebaTe tool depends on the number of cores of the system. For the studied benchmarks, the average execution time is 9.3 minutes in a 1.3GHz AMD processor with 512MBytes of RAM memory. Benchmarks q12710 and g1023 require, respectively, the smallest and the largest execution time. As for the NoCBaTe tool, the execution time depends on the number of test packets, but it is much smaller than the other method. System p34392 is the most expensive one, requiring 1.81 seconds to find a solution.

Finally, the results of Table 6.1 demonstrate how the proposed test planning methods can be included in the beginning of the system design cycle. Indeed, both methods can be applied to a project as soon as the system functionality and the initial set of cores is chosen. With the basic information about the cores the system integrator can define a possible floorplanning for the chip and use the ReBaTe tool to check the possible test costs for a core-to-core connection model. Similarly, the NoCBaTe tool can be executed assuming a communication platform is used. Thus, for example, if the designer is choosing between a bus-based or a NoC-based system, the test costs can be used as another parameter in the evaluation of both solutions, even before the system synthesis. Moreover, the designer can also verify how different versions of the same functionality impact the system test or inform a core provider how to improve the test of a block according to the system constraints.

6.2 Reuse-based versus Bus-based Test Planning

Some recent works have been using the ITC'02 benchmarks to evaluate different test planning methods. However, the solutions proposed by those works is usually based on the insertion of test buses. In fact, there is no other recent work that defines a reuse-based TAM and uses the mentioned benchmarks as case studies, to the best knowledge of this author.

Despite the distinct nature of the bus-based test planning approaches compared to the reuse-based ones, and considering the several assumptions made over the benchmark descriptions for its use in the proposed test planning tools, one can still compare, at least partially, the reuse-based techniques described in this work with the most recent literature.

The following methods will be used in this comparison:

- **[1]** refers to (IYENGAR; CHAKRABARTY; MARINISSEN, 2002b), where a method for wrapper/TAM co-optimization considering variable-width test buses is presented. The problem is modeled as a rectangle packing or two-dimensional packing. Firstly,

the wrapper of each core in the system is optimized to find the best configuration between the core's test data requirements and its TAM width. Then, a test schedule is determined so that an effective amount of TAM width is assigned to each core. At last, the wrapper corresponding to the TAM width assigned to the core is chosen. The results for four benchmarks are presented in that work: d695, p22810, p34392, and p93791.

- **[2]** refers to (HUANG et al., 2002). In that work, the wrapper and TAM co-optimization under power constraints is modeled as a restricted 3-D bin packing problem. The method allows, the selection of optimal wrapper width for each core, the allocation of SOC pins to cores, and the scheduling of core tests to achieve minimal test times.

- **[3]** refers to the work of Goel and Marinissen (GOEL; MARINISSEN, 2002c), where test architectures are defined for a SoC in such a way that the required ATE vector memory depth and the system test time are minimized. The algorithm proposed in that work efficiently determines the number of of TAMs and their widths, the assignment of modules to TAMs, and the wrapper design per module. The results for all twelve benchmarks are presented for a number of TAM architectures. That work also defines an equation to evaluate the lower bound for the test time of an SoC for bus-based TAMs with fixed-width (independent of the architecture).

- **[4]** refers to a test scheduling method based on the simulated annealing algorithm proposed in (ZOU et al., 2003). The test scheduling problem is also formulated as a two-dimensional bin packing problem (rectangle packing) and a data structure called a sequence pair is used to represent the placement of the rectangles. Then, simulated annealing is used to find the optimal test schedule by altering an initial sequence pair and changing the width of the core wrapper. The results for all ITC'02 benchmarks are provided in that work.

Table 6.2 presents the comparative results for the ITC'02 benchmarks. For the bus-based methods, the solutions that use a total TAM bitwidth of 32 pins are presented. The lower bound for the test time calculated in (GOEL; MARINISSEN, 2002c) is presented in the first column. The next four columns present the test time for each of the bus-based method considered. The results for the ReBaTe tool are shown in column six, with the number of extra pins for the solution informed in parenthesis. Finally, the results for the NoC-based method are shown in the last column, for a network of 32 bits.

For the ReBaTe tool, only the solutions for pins, time, and area optimization are considered, since power-aware results are not presented for the other methods. Among the solutions found by the tool, the ones with smaller test times and a number of extra pins less or equal 32 were selected. From [2], the results for power unrestricted systems are selected and from [3], the results for the Test Rail architecture with serial scheduling are used.

The area overhead of the ReBaTe solutions is 5% for the two synthetic benchmarks from Duke University (d695 and d281). For the remaining benchmarks, the average area overhead is 1.3%.

From Table 6.2 the following observations can be made:

- For some systems, the comparison among the bus-based and the reuse-based methods is not possible. In the bus-based methods, the BISTed modules are not considered in the evaluation of the test time, since they aim at the configuration of the

Table 6.2: Comparative results with bus-based methods for W = 32

Benchmark	Lower bound [3]	[1]	[2]	[3]	[4]	ReBaTe	NoCbaTe 32-bit NoC
d695	20,482	23,021	42,716	21,690	41,604	16,566 (8)	41,764
g1023	15,088	–[1]	31,444	16,855	31,139	38,773 (21)	27,878
f2126	335,334	–[1]	357,109	335,334	357,088	383,427 (21)	565,261
q12710	2,222,349	–[1]	2,222,349	2,222,349	2,222,349	2,222,349 (21)	6,230,935
t512505	5,228, 420	–[1]	10,531,003	5,268,868	10,530,995	5,857,190 (23)	5,649,467
h953	119,357	–[1]	122,636	119,357	119,357	119,357 (21)	231,278
u226[2]	6,992	–[1]	13,416	18,663	13,333	1,363,968 (3)	1,392,151
d281[2]	3,926	–[1]	7,948	4,163	7,432	8,648 (8)	9,024
p22810	209,734	246,150	446,684	226,640	438,619	281,626 (76)	371,814
p34392	544,579	544,579	1,016,640	552,746	944,768	791,185 (30)	1,468,113
p93791	873,334	975,016	1,791,860	940,745	1,757,452	530,667 (77)	449,422
a568710[2]	18,947,848	–[1]	42,198,943	22,475,033	32,626,782	15,726,859 (12)	–[1]

[1] result not available

[2] SoCs with BISTed modules

cores scan chains to minimize the core test time. This is the case, for example, of systems d281, u226, and a586710. The papers describing the bus-based methods do not specify whether the cores that do not use scan chains are optimized. It is assumed the test time of those modules is indeed considered.

- The bus-based methods considered in Table 6.2 fully optimize the wrapper of each core in the system, as well as the usage of the available TAM wires. The reuse-based methods, on the other hand, optimize the wrappers according to the available functional connections and the available time slot in the test schedule. Moreover, the global optimization of the system costs is the goal of the proposed methods, rather than the total minimization of the test time alone. Despite this important difference in objectives, one can observe that the ReBaTe tool generates test solutions with a test time that is in accordance (same or smaller magnitude) to the test times of the other methods, for a very low cost in terms of pins and area. The time overhead of the ReBaTe tool is very high for system g1023 (157%), but is at most 45% for the remaining benchmarks, with respect to the lower bound of the bus-based methods. However, in two cases this overhead is negative (systems d695 and a586710). Notice that system a586710 has some BISTed cores that are not considered in the bus-based methods. Considering the best solution found by the bus-based approaches, this overhead is practically the same.

- As for the NoCBaTe tool, the results in terms of test time are also in accordance with the other methods, although the time overhead is usually higher than the bus-based lower bound. This is mainly caused by the reuse of the available system interfaces and the definition of the test packets. Indeed, one can observe that the systems with worse test times in the NoCBaTe tool are the ones with a reduced number of modules that are actually connected to the network. This is the case, for example, of systems f2126, q12710, and p34392, that have only four modules connected at the highest level of hierarchy. System h953 has nine cores connected to the network, but a very reduced number of input and output pins (enough for only 1 input and 1 output ports), which does not allow the usage of the network parallelism. One can consider this example as a system with a single test bus shared among all cores. In addition, systems d695, g1023, and p22810 have a test time that is comparable to the results of at least one bus-based method, whereas system p93701 presents a gain in the test time of 48% over the bus-based lower bound.

- As mentioned in Chapter 4, the solutions provided by the ReBaTe method strongly depend on the system characteristics, such as the size of the cores and the chip placement. Therefore, the solution devised for the actual description of the ITC'02 benchmarks can be different (better or worse) from the ones presented in Table 6.2. However, from the comparative results presented for system u226 in Chapter 4, one can expect that better results can be achieved by this method.

- The test time given by the NoCbaTe approach is usually higher, but in the same order of magnitude of the bus-based methods. However, one must consider that the NoC-based method implies no pins and a minimal area overhead, and presents no further synthesis efforts besides the wrappers modifications. Hence, for such a low-cost technique, the test time can be considered very competitive. Again, for the actual system, a different number of interfaces with the tester can be considered, which can improve the presented results.

6.3 Limitations of the Proposed Methods

From the results presented in Sections 6.1 and 6.2, some limitations of the proposed methods can be identified:

- for the ReBaTe tool, the search space is defined by a heuristic that establishes a new test time limit for which a TAM is devised. The use of a partial solution to define the next test time is used in the experiments presented in this text. However, for some systems, another heuristic led to a different design space which resulted in better results. Thus, the study of other heuristics for the definition of the test time limit and to avoid the local minima is required. Moreover, the execution time of this method is also defined by this heuristic, which may be a problem for larger systems.

- In general, the reuse of the system resources is the main advantage and the main disadvantage of the proposed solutions. Although the RebaTe technique is capable of inserting new resources in the system when necessary, this option is avoided as much as possible to reduce area and pins costs, thus imposing a stronger penalty in the system test time. Similarly, for the NoC-based method, the test time is defined for a fixed number of interfaces, defined by the system application. Thus, the search space is quite reduced and it is up to the designer to insert other interfaces and generate new solutions.

- The optimization of the wrapper of each module is not applied in the proposed techniques. However, this procedure is an important point for the test time minimization of the bus-based methods, and does not imply any modification or overhead at system level. Therefore, the application of such a mechanism for the minimization of the core test time before optimizing the system aspects can improve the results in both, the ReBaTe and the NoCBaTe tools.

- For the NoC-based test, the definition of the test packet in conjunction to the wrapper optimization can improve even further the system test time. For example, the total capacity of the channel is not always used in the current test packet model, since each bit of the core scan chains must be carried by subsequent flits in the test packet. One can also consider the impact of carrying more than one vector per packet, for instance.

- Additionally, it is important to consider system interfaces with more or less pins than the communication channel of the network, so that all system pins can be reused. Moreover, the possibility of inclusion of extra pins should also be considered by the NoCBaTe tool. In this case, the location and number of extra pins must be also devised by the tool.

- At-speed test is not directly addressed in this work, and should be considered in both techniques. Similarly, the test of the interconnections is not considered, under the assumption that the reuse of the available connections can only simplify the test of these structures. However, this issue must also be evaluated in more detail, as well as the test of the on-chip network before its reuse.

Overall, the proposed methods can be used to estimate the system test costs in the first steps of the system design. The choice for one or another method can thus be based on the system constraints and characteristics, rather than only on the core requirements.

Although the main goal of this work was to find a test solution with a good trade-off among the several test and synthesis costs, the test time of the solutions devised by both methods is comparable to the results of the most recent bus-based techniques, whose main goal is to minimize this variable only. Despite the positive results presented in this thesis, some improvements in the proposed techniques should still be considered.

7 FINAL REMARKS

This thesis presented two innovative approaches for the test planning of core-based systems. Considering the recent advances of the electronic design with respect to the use of pre-designed blocks as cores or platforms, the main problems related to the test of the new complex systems were described in three levels: the core test, the interconnect test, and the system-level test. Then, a number of solutions for some of these problems were discussed. From the studied solutions, it was possible to identify that most techniques aim at efficiently define a test access mechanism to each embedded core, and, based on this mechanism, define an efficient test schedule. However, the system-level aspects of the test and its inclusion in the earlier design steps had not been considered yet. Hence, the optimization of the test costs together with the system synthesis was defined as the main goal of this work. Responding to this goal, two test planning methods were proposed to help the designer to estimate the test costs of the system as early as possible. In addition, both techniques rely on the reuse of the system resources during test as a means of reducing the system test costs.

The first proposed method was presented in Chapter 4 and is based on the definition of multiple TAMs inside the chip and on a fine-grained search algorithm for the exploration of the design space. The multi-TAM model considers the reuse of functional connections as well as other access mechanisms, such as partial test buses, core transparency modes, and other bypass modes. The test schedule is defined in conjunction with the access mechanism so that good trade-offs among the costs of pins, area, and test time can be sought. Furthermore, system power constraints can also be considered for the definition of the test solution. Experimental results have shown that different trade-offs can be achieved for a variety of system constraints. They have also shown that the reuse can lead to a very cost-effective test solution, even when several test cost factors are optimized. On the other hand, the proposed method is very dependent not only on the available functional connections, but also on the placement of the cores inside the chip and on the number of available pins in the system interface. Current works include the definition of the core neighborhood based on the placement, in order to reduce the search space and speed up the method. Moreover, the use of the complete wrapper co-optimization method proposed in (IYENGAR; CHAKRABARTY; MARINISSEN, 2001) can improve the results for those systems whose functional connections make the reuse more difficult (several feedbacks, functional connections with reduced bitwidth, etc).

In Chapter 5, the reuse of on-chip networks for the test of core-based systems was discussed. A power-aware test scheduling technique based on the list-scheduling algorithm was proposed to minimize the system test time when the network and the system interfaces are reused. The required modifications in the wrappers that connect the cores to the communication platform were also explained. Experimental results for the ITC'02

SoC Test benchmarks were presented and discussed, for a number of possible implementations of those systems in a NoC. Those results have shown that when the test tasks are scheduled a priori over the available network resources, system test times are comparable to other techniques where exclusive TAMs are used. The main advantage of the proposed technique is the possibility of obtaining reduced test times with minimum pin and area overhead. Moreover, the reuse provides cheaper test methods, since the interconnections can be tested by the same technique. Current works include the definition of a more accurate model for the power consumption evaluation. Other network topologies, such as fat-tree, with non-deterministic routing algorithms, can also be studied.

In Chapter 6 the proposed techniques were compared to each other and to other recent test planning approaches. Despite the limitations of the reuse-based methods listed in that chapter, the new methods present a good performance in terms of test time even considering the optimization of more than one cost variable.

Some technical contributions of this work can be mentioned:

1. the expansion of concerns during the test planning of core-based systems. Most works in this area consider a single access mechanism and aim at minimizing the system test time, without considering the system characteristics. In the first test planning approach defined in this thesis, the whole system is considered and a solution that represents a good compromise among a number of costs is sought. This method was firstly published in the *Design, Automation, and Test in Europe (DATE)* conference, in 2002 (COTA et al., 2002). An improved version of the ReBaTe tool was published in the *Latin-American Test Workshop (LATW)* of 2002 (COTA et al., 2002) and was recently accepted to the *Journal of Electronic Testing: Theory and Applications (JETTA)* (COTA et al., 2003);

2. the comparison of reuse-based and bus-based methods. With the currently SoC test benchmarks, it was possible to establish some rules and metrics for comparison of test planning methods. Experimental results presented in Chapters 4 and 6 show that the use of the current benchmarks for comparison among test planning approaches should be carefully considered, since different assumptions can generate very different solutions. This discussion was presented in the *Latin-American Test Workshop (LATW)* of 2003 (COTA; CARRO; LUBASZEWSKI, 2003a);

3. the reuse of the on-chip networks as test access mechanism. This is the first work that considers the test of heterogeneous NoC-based systems and reuses the available communication platform. The method was firstly published in the *VLSI Test Symposium (VTS)* of 2003 (COTA et al., 2003). The power-aware test scheduling was presented at the *European Test Workshop (ETW)* of 2003 (COTA et al., 2003a) and was accepted to the *International Test Conference (ITC)* of 2003 (COTA et al., 2003b);

4. although not presented in this text, the definition of an embedded test controller was also a result of this work. The **M**icroprocessor for **E**mbedded **T**est or MET was primarily developed to be used by the cores that require at-speed testing. It was firstly presented at the *5th IEEE Workshop on Testing Embedded Core-based Systems* (COTA et al., 2001), and later improved in (CASSOL, 2002).

Despite the good performance of the proposed test planning techniques, there are some open points that can be studied in the sequel of this work:

1. the study of other heuristics to establish the search space of the ReBaTe method;
2. the inclusion of the MET controller as a test resource in the ReBaTe approach;
3. the optimization of the wrappers before defining the access mechanism for each core, as proposed in (IYENGAR; CHAKRABARTY; MARINISSEN, 2001) is another modification that can improve the results of both methods proposed in this work;
4. the implementation of one of the benchmarks for the validation of the equations that define the area overhead and simulation of the devised test solution. Currently, benchmark d695 is being implemented in VHDL along with a test solution devised by the ReBaTe tool;
5. the study of the automatic generation of the P1500 wrapper modes required by the TAM defined in the ReBaTe approach, to make the implementation of the test solution easier. Moreover, this study must consider the requirements in terms of control signals and configuration time of those wrappers;
6. the consideration of other network topologies in the NoCbaTe method;
7. the verification of the actual impact of system interfaces with smaller bitwidth than the network channel on the test time;
8. the verification of the system test time and area overhead for other models of test packets and other routing algorithms, such as dedicated routing. Moreover, the use of compression techniques in conjuntion with the wrapper modification can be an interesting alternative to further optimize the test time of a NoC-based system;
9. although the results of this work were generated for a realistic but estimated power consumption model, the refinement of the power profile of the network, considering communication channels of different lengths, for example, is an important issue. In addition, more detailed power consumption models for the embedded cores, as the one proposed in (ROSINGER; AL-HASHIMI; NICOLICI, 2002), can also be considered in the test scheduling algorithm;

Finally, considering this research subject, more general issues can be considered as future works:

- the inclusion of the test of analog blocks in the test planning methods. BISTed analog blocks may present higher test times than their digital counterparts, and this time can be used to reduce the extra digital test resources. Moreover, some test approaches for analog blocks being studied in this research group (NEGREIROS; CARRO; SUSIN, 2003) are based on the reuse of digital blocks of the SoC. Therefore, the test scheduling of both, analog and digital blocks must be defined in conjunction;
- BISTed cores are considered a powerful solution to reduce the system and ATE requirements in terms of access mechanisms. However, the impact of such blocks in the test planning has not been extensively studied. Preliminary results considering the definition of the best set of BISTed cores in the NoC-based method have been presented in the *Test Resource Partitioning (TRP)* workshop of 2003 (COTA;

CARRO; LUBASZEWSKI, 2003b). However, a more detailed study on this subject is necessary, as well as its inclusion in the ReBaTe approach.

REFERENCES

AERTS, J.; MARINISSEN, E. Scan Chain Design for Test Time Reduction in Core-based ICs. In: INTERNATIONAL TEST CONFERENCE, 1998. **Proceedings...** Los Alamitos: IEEE Computer Society, 1998. p.448–457.

ALLIANCE, V. **IEEE P1450 Web Site**. [S.l.: s.n.], 2003. Available at: <http://grouper.ieee.org/groups/1450/>. Acessed in: August 2003.

ALLIANCE, V. **VSI Alliance Web Site**. [S.l.: s.n.], 2003. Available at: <http://www.vsi.org/>. Acessed in: August 2003.

BABB, J.; RINARD, M.; MORITZ, C.; LEE, W.; FRANK, M.; BARUA, R.; AMARASINGHE, S. Parallelizing Applications Into Silicon. In: ANNUAL IEEE SYMPOSIUM ON FIELD-PROGRAMMABLE CUSTOM COMPUTING MACHINES, 7., 1999. **Proceedings...** [S.l.: s.n.], 1999. p.70–80.

BASU, S.; MUKHOPADHAY, D.; ROYCHOUDHURY, D.; SENGUPTA, I.; BHAWMIK, S. Reformatting Test Patterns for Testing Embedded Core Based System Using Test Access Mechanism (TAM) Switch. In: ASIA AND SOUTH PACIFIC DESIGN AUTOMATION CONFERENCE, 7., 2002. **Proceedings...** [S.l.: s.n.], 2002. p.598–603.

BASU, S.; SENGUPTA, I.; CHOWDHURY, D. R.; BHAWMIK, S. An Integrated Approach to Testing Embedded Cores and Interconnects Using Test Access Mechanism (TAM) Switch. **Journal of Electronic Testing: Theory and Applications**, The Netherlands, v.18, n.4, p.475–485, Aug. 2002.

BENABDENBI, M.; MAROUFI, W.; MARZOUKI, M. Cas-Bus: A Scalable and Reconfigurable Test Acces Mechanism for Systems on a Chip. In: DESIGN, AUTOMATION AND TEST IN EUROPE CONFERENCE, 2000, Paris, FR. **Proceedings...** Los Alamitos: IEEE Computer Society, 2000. p.141–145.

BENABDENBI, M.; MAROUFI, W.; MARZOUKI, M. CAS-BUS: A Test Access Mechanism and a Toolbox Environment for Core-Based System Chip Testing. **Journal of Electronic Testing: Theory and Applications**, The Netherlands, v.18, n.4, p.455–473, Aug. 2002.

BENINI, L.; MICHELI, G. D. Networks on Chips: a New SOC Paradigm. **IEEE Computer**, [S.l.], v.35, n.1, p.70–78, Jan. 2002.

BENSO, A.; CHIUSANO, S.; DI CARLO, S.; PRINETTO, P.; RICCIATO, F.; SPADARI, M.; ZORIAN, Y. HD2BIST: a Hierarchical Framework for BIST Scheduling, Data Patterns Delivering and Diagnosis in SoCs. In: INTERNATIONAL TEST CONFERENCE, 2000. **Proceedings. . .** Los Alamitos: IEEE Computer Society, 2000. p.892–901.

BERGAMASCHI, R.; BHATTACHARYA, S.; WAGNER, R.; FELLENZ, C.; MUHLADA, M.; WHITE, F.; DAVEAU, J.-M.; LEE, W. Automating the Design of SOCs Using Cores. **IEEE Design & Test of Computers**, [S.l.], v.18, n.5, p.32–45, Sep./Oct. 2001.

BERGAMASCHI, R.; COHN, J. The A to Z of SoCs. In: IEEE/ACM INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN, 2002. **Proceedings. . .** [S.l.: s.n.], 2002. p.791–798.

BHATTACHARYA, D. Hierarchical Test Access Architecture for Embedded Cores in an Integrated Circuit. In: IEEE VLSI TEST SYMPOSIUM, 16., 1998. **Proceedings. . .** Los Alamitos: IEEE Computer Society, 1998. p.8–14.

BOARD, I. S. **IEEE 1450.0**:Standard Test Interface Language (STIL) for Digital Test Vector Data - Language Manual. New York: [s.n.], 1999.

CASSOL, L. J. **Teste de Sistemas Integrados Utilizando Controladores Específicos**. 2002. M.Sc. Thesis — (Programa de Pós-Graduação em Engenharia Elétrica) - Escola de Engenharia, UFRGS, Porto Alegre.

CHAKRABARTY, K. Test Scheduling for Core-Based Systems. In: INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN, 1999. **Proceedings. . .** [S.l.: s.n.], 1999. p.391–394.

CHAKRABARTY, K. Design of System-on-a-chip Test Access Architectures Using Integer Linear Programming. In: IEEE VLSI TEST SYMPOSIUM, 18., 2000. **Proceedings. . .** Los Alamitos: IEEE Computer Society, 2000. p.127–134.

CHAKRABARTY, K. Design of System-on-a-Chip Test Access Architectures Under Place-and-Route and Power Constraints. In: ACM/IEEE DESIGN AUTOMATION CONFERENCE, 2000, Los Angeles, USA. **Proceedings. . .** New York: ACM, 2000. p.432–437.

CHAKRABARTY, K. Test Scheduling for Core-based Systems Using Mixed-integer Linear Programming. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, [S.l.], v.19, n.10, p.1163–74, Oct. 2000.

CHAKRABARTY, K.; MUKHERJEE, R.; A., E. Synthesis of Transparent Circuits for Hierarchical and System-on-a-chip Test. In: INTERNATIONAL CONFERENCE ON VLSI DESIGN, 14., 2001. **Proceedings. . .** [S.l.: s.n.], 2001. p.431–436.

CHAKRABORTY, T.; BHAWMIK, S.; CHIANG, C.-H. Test Access Methodology for System-on-chip Testing. In: IEEE WORKSHOP ON TESTING EMBEDDED CORE-BASED SYSTEMS, 4., 2000, Los Angeles, USA. **Digest of Papers. . .** [S.l.: s.n.], 2000. p.1.1–1–1.1–7.

CHANDRA, A.; CHAKRABARTY, K. Reduction of SOC Test Data Volume, Scan Power and Testing Time Using Alternating Run-length Codes. In: ACM/IEEE DESIGN AUTOMATION CONFERENCE, 2002, New Orleans, USA. **Proceedings. . .** New York: ACM, 2002. p.673–678.

CHANDRA, A.; CHAKRABARTY, K. Test data compression and decompression based on internal scan chains and Golomb coding. **IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems**, [S.l.], v.21, n.6, p.715 –722, June 2002.

CHANDRA. Simultaneous Module Selection and Scheduling for Power-constrained Testing of Core Based Systems. In: INTERNATIONAL CONFERENCE ON VLSI DESIGN, 2000. **Proceedings. . .** [S.l.: s.n.], 2000. p.462–467.

CHATTOPADHYAY S.; REDDY, K. Genetic Algorithm Based Test Scheduling and Test Access Mechanism Design for System-on-chips. In: INTERNATIONAL CONFERENCE ON VLSI DESIGN, 16., 2003. **Proceedings. . .** [S.l.: s.n.], 2003. p.341–346.

CHEN, L.; BAI, X.; DEY, S. Testing for Interconnect Crosstalk Defects Using On-Chip Embedded Processor Cores. **Journal of Electronic Testing: Theory and Applications**, The Netherlands, v.18, n.4, p.529–538, Aug. 2002.

CHIUSANO, S.; PRINETTO, P.; WUNDERLICH, H.-J. Non-Intrusive BIST for Systems-on-a-chip. In: INTERNATIONAL TEST CONFERENCE, 2000. **Proceedings. . .** Los Alamitos: IEEE Computer Society, 2000. p.644–651.

CHOU, R.; SALUJA, K.; AGRAWAL, V. Scheduling Tests for VLSI Systems Under Power Constraints. **IEEE Transactions on VLSI**, [S.l.], v.5, n.2, p.175–184, June 1997.

CMP. **Design and Reuse**: the Catalyst of Collaborative SoC Design Through SIP Exchange. [S.l.: s.n.], 2003. Available at: <http://www.eedesign.com/sip/>. Acessed in: July 2003.

COTA, E.; BRISOLARA, L.; CARRO, L.; SUSIN, A.; LUBASZEWSKI, M. MET: A Microprocessor for Embedded Test. In: IEEE WORKSHOP ON TESTING EMBEDDED CORE-BASED SYSTEMS, 5., 2001, Los Angeles, USA. **Digest of Papers. . .** [S.l.: s.n.], 2001.

COTA, E.; CARRO, L.; LUBASZEWSKI, M. SOC Benchmarks and System Test Planning: How Much Information is Enough? In: LATIN AMERICAN TEST WORKSHOP, 2003, Natal, Brazil. **Digest of Papers. . .** [S.l.: s.n.], 2003.

COTA, E.; CARRO, L.; LUBASZEWSKI, M. BISTed Cores and Test Time Minimization in NOC-based Systems. In: IEEE INTERNATIONAL WORKSHOP ON TEST RESOURCE PARTITIONING, 2003, Napa Valley, USA. **Digest of Papers. . .** [S.l.: s.n.], 2003. p.14–19.

COTA, E.; CARRO, L.; LUBASZEWSKI, M.; ORAILOGLU, A. Generic and Detailed Search for TAM Definition in Core-based Systems. In: LATIN AMERICAN TEST WORKSHOP, 2002. **Digest of Papers. . .** [S.l.: s.n.], 2002. p.160–164.

COTA, E.; CARRO, L.; LUBASZEWSKI, M.; ORAILOGLU, A. Achieving Global Test Costs Optimization in Core-based Systems. **Accepted to the Journal of Electronic Testing: Theory and Applications**, [S.l.], 2003.

COTA, E.; CARRO, L.; ORAILOGLU, A.; LUBASZEWSKI, M. Test Planning and Design Space Exploration in Core-based Environment. In: DESIGN, AUTOMATION AND TEST IN EUROPE, 2002, Paris, FR. **Proceedings...** Los Alamitos: IEEE Computer Society, 2002. p.483–490.

COTA, E.; CARRO, L.; WAGNER, F.; LUBASZEWSKI, M. Power-aware NoC Reuse on the Testing of Core-based Systems. In: EUROPEAN TEST WORKSHOP, 2003, Maastricht, NL. **Digest of Papers...** [S.l.: s.n.], 2003. p.123–128.

COTA, E.; CARRO, L.; WAGNER, F.; LUBASZEWSKI, M. Power-aware NoC Reuse on the Testing of Core-based Systems. In: INTERNATIONAL TEST CONFERENCE, 2003, Charlotte, USA. **Proceedings...** Los Alamitos: IEEE Computer Society, 2003. p.612–621.

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

DALLY, W. J.; TOWLES, B. Route Packets, Not Wires: On-Chip Interconnection Networks. In: DESIGN AUTOMATION CONFERENCE, 2001. **Proceedings...** New York: ACM, 2001. p.684–689.

DUATO, J.; YALAMANCHILI, S.; NI, L. **Interconnection Networks**: an Engineering Approach. Los Alamitos: IEEE Computer Society, 1997.

EBADI, Z.; IVANOV, A. Design of an Optimal Test Access Architecture Using a Genetic Algorithm. In: ASIAN TEST SYMPOSIUM, 2001. **Proceedings...** [S.l.: s.n.], 2001. p.205–210.

FLOTTES, M.-L.; POUGET, J.; ROUZEYRE, B. A Heuristic for Test Scheduling at System Level. In: DESIGN, AUTOMATION AND TEST IN EUROPE CONFERENCE, 2002, Paris, FR. **Proceedings...** Los Alamitos: IEEE Computer Society, 2002. p.1124.

FORSELL, M. A Scalable High-performance Computing Solution for Networks on Chips. **IEEE Micro**, [S.l.], v.22, n.5, p.46–55, Sep./Oct. 2002.

GEREZ, S. H. **Algorithms for VLSI Design Automation**. [S.l.]: Baffins Lane, Chichester, England, John Wiley & Sons, 1998.

GHOSH, I.; DEY, S.; JHA, N. A Fast and Low Cost Testing Technique for Core-based System-on-chip. In: ACM/IEEE DESIGN AUTOMATION CONFERENCE, 1998. **Proceedings...** New York: ACM, 1998. p.542–547.

GHOSH, I.; JHA, N.; DEY, S. A Low Overhead Design for Testability and Test Generation Technique for Core-based Systems. In: INTERNATIONAL TEST CONFERENCE, 1997, Washington, USA. **Proceedings...** Los Alamitos: IEEE Computer Society, 1997. p.50–59.

GOEL, S.; MARINISSEN, E. Cluster-based Test Architecture Design for System-on-chip. In: IEEE VLSI TEST SYMPOSIUM, 2002. **Proceedings...** Los Alamitos: IEEE Computer Society, 2002. p.259–264.

GOEL, S.; MARINISSEN, E. A Novel Test Time Reduction Algorithm for Test Architecture Design for Core-based System Chips. In: IEEE EUROPEAN TEST WORKSHOP, 7., 2002. **Proceedings...** Los Alamitos: IEEE Computer Society, 2002. p.7–12.

GOEL, S.; MARINISSEN, E. Effective and Efficient Test Architecture Design for SOCs. In: INTERNATIONAL TEST CONFERENCE, 2002. **Proceedings...** Los Alamitos: IEEE Computer Society, 2002. p.529–538.

GONCIARI, P.; AL-HASHIMI, B.; NICOLICI, N. Integrated Test Data Decompression and Core Wrapper Design for Low-cost System-on-a-chip Testing. In: INTERNATIONAL TEST CONFERENCE, 2002. **Proceedings...** Los Alamitos: IEEE Computer Society, 2002. p.64–73.

GUERRIER, P.; GREINER, A. A Generic Architecture for On-Chip Packet-Switched Interconnections. In: DESIGN, AUTOMATION AND TEST IN EUROPE, 2000, Paris, FR. **Proceedings...** Los Alamitos: IEEE Computer Society, 2000. p.250–256.

GUPTA, R. K.; ZORIAN, Y. Introducing Core-Based System Design. **IEEE Design & Test of Computers**, [S.l.], v.13, n.4, p.15–25, Oct./Dec. 1997.

HALES, A.; MARINISSEN, E. J. **IEEE P1500 Web Site**. [S.l.: s.n.], 2003. Available at: <http://grouper.ieee.org/groups/1500/>. Acessed in: August 2003.

HU, H.; YIBE, S. A Scalable Test Mechanism and its Optimization for Test Access to Embedded Cores. In: INTERNATIONAL CONFERENCE ON ASIC, 4., 2001. **Proceedings...** [S.l.: s.n.], 2001. p.773–776.

HUANG, Y.; CHENG, W.-T.; TSAI, C.-C.; MUKHERJEE, N.; SAMMAN, O.; ZAIDAN, Y.; REDDY, S. M. Resource Allocation and Test Scheduling for Concurrent Test of Core-based SoC Design. In: ASIAN TEST SYMPOSIUM, 2001. **Proceedings...** Los Alamitos: IEEE Computer Society, 2001. p.265–270.

HUANG, Y.; CHENG, W.-T.; TSAI, C.-C.; MUKHERJEE, N.; SAMMAN, O.; ZAIDAN, Y.; REDDY, S. M. On Concurrent Test of Core-Based SOC Design. **Journal of Electronic Testing: Theory and Applications**, The Netherlands, v.18, n.4, p.401–414, Aug. 2002.

HUANG, Y.; REDDY, S.; CHENG, W.-T.; REUTER, P.; MUKHERJEE, N.; TSAI, C.-C.; SAMMAN, O.; ZAIDAN, Y. Optimal Core Wrapper Width Selection and SOC Test Scheduling Based on 3-D Bin Packing Algorithm. In: INTERNATIONAL TEST CONFERENCE, 2002. **Proceedings...** Los Alamitos: IEEE Computer Society, 2002. p.74–82.

HWANG, S.; ABRAHAM, J. Reuse of Addressable System Bus for SOC Testing. In: ANNUAL IEEE INTERNATIONAL ASIC/SOC CONFERENCE, 14., 2001. **Proceedings...** [S.l.: s.n.], 2001. p.215–219.

IEEE Standards Board. **IEEE 1149.1**:IEEE Standard Test Access Port and Boundary Scan Architecture. New York, 1990.

IYENGAR, V. Precedence-based, Preemptive, and Power-constrained Test Scheduling for System-on-a-chip. In: IEEE VLSI TEST SYMPOSIUM, 2001, Los Angeles, USA. **Proceedings...** Los Alamitos: IEEE Computer Society, 2001. p.368–374.

IYENGAR, V.; CHAKRABARTY, K. Iterative Test Wrapper and Test Access Mechanism Co-optimization. In: IEEE WORKSHOP ON TESTING EMBEDDED CORE-BASED SYSTEMS, 5., 2001, Los Angeles, USA. **Digest of Papers...** [S.l.: s.n.], 2001.

IYENGAR, V.; CHAKRABARTY, K. System-on-a-chip Test Scheduling With Precedence Relationships, Preemption, and Power Constraints. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, [S.l.], v.21, n.9, p.1088–1094, Sept. 2002.

IYENGAR, V.; CHAKRABARTY, K.; MARINISSEN, E. Recent Advances in Test Planning for Modular Testing of Core-based SOCs. In: ASIAN TEST SYMPOSIUM, 2002. **Proceedings...** Los Alamitos: IEEE Computer Society, 2002. p.320–325.

IYENGAR, V.; CHAKRABARTY, K.; MARINISSEN, E. On Using Rectangle Packing for SOC Wrapper/TAM Co-optimization. In: IEEE VLSI TEST SYMPOSIUM, 2002. **Proceedings...** Los Alamitos: IEEE Computer Society, 2002. p.253–258.

IYENGAR, V.; CHAKRABARTY, K.; MARINISSEN, E. Wrapper/TAM Co-optimization, Constraint-driven Test Scheduling, and Tester Data Volume Reduction for SOCs. In: ACM/IEEE DESIGN AUTOMATION CONFERENCE, 2002, New Orleans, USA. **Proceedings...** New York: ACM, 2002. p.685–690.

IYENGAR, V.; CHAKRABARTY, K.; MARINISSEN, E. J. Iterative Test Wrapper and Test Access Mechanism Co-optimization. In: INTERNATIONAL TEST CONFERENCE, 2001, Baltimore, USA. **Proceedings...** Los Alamitos: IEEE Computer Society, 2001. p.1023–1032.

IYENGAR, V.; CHAKRABARTY, K.; MARINISSEN, E. J. Test Wrapper and Test Access Mechanism Co-Optimization for System-on-Chip. **Journal of Electronic Testing: Theory and Applications**, The Netherlands, v.18, n.2, p.213–230, Apr. 2002.

IYENGAR, V.; CHANDRA, A.; SCHWEIZER, S.; CHAKRABARTY, K. A Unified Approach for SoC Testing Using Test Data Compression and TAM Optimization. In: DESIGN, AUTOMATION AND TEST IN EUROPE CONFERENCE, 2003. **Proceedings...** Los Alamitos: IEEE Computer Society, 2003. p.1188–1189.

JERVAN, G.; PENG, Z.; UBAR, R. Test Cost Minimization for Hybrid BIST. In: IEEE INTERNATIONAL SYMPOSIUM ON DEFECT AND FAULT TOLERANCE IN VLSI SYSTEMS, 2000. **Proceedings...** [S.l.: s.n.], 2000. p.283–291.

JERVAN, G.; PENG, Z.; UBAR, R.; KRUUS, H. A Hybrid BIST Architecture and its Optimization for SoC Testing. In: INTERNATIONAL SYMPOSIUM ON QUALITY ELECTRONIC DESIGN, 2002. **Proceedings...** [S.l.: s.n.], 2002. p.273 –279.

KAPUR, R.; KELLER, B.; KOENEMANN, B.; LOUSBERG, M.; REUTER, P.; TAYLOR, T.; VARMA, P. P1500-CTL: Towards a Standard Core Test Language. In: VLSI TEST SYMPOSIUM, 1999. **Proceedings...** Los Alamitos: IEEE Computer Society Press, 1999. p.489–490.

KAPUR, R.; LOUSBERG, M.; TAYLOR, T.; KELLER, B.; REUTER, P.; KAY, D. CTL the Language for Describing Core-based Test. In: INTERNATIONAL TEST CONFERENCE, 2001. **Proceedings...** Los Alamitos: IEEE Computer Society, 2001. p.131–139.

KAPUR, R.; MARINISSEN, E.; MUKHERJEE, N.; RICCHETI, M.; TAYLOR, T.; UDELL, J. **P1500/D0.3**:Draft Standard for Embedded Core Test (SECT). [S.l.]: IEEE P1500 Documentation Task Force, 2001. Preliminar draft standard.

KARIM, F.; NGUYEN, A.; DEY, S. An Interconnect Architecture for Networking Systems on Chips. **IEEE Micro**, [S.l.], v.22, n.5, p.36–45, Sep./Oct. 2002.

KEUTZER, K.; NEWTON, A.; RABAEY, J.; SANGIOVANNI-VINCENTELLI, A. System-Level Design: Orthogonalization of Concerns and Platform-Based Design. **IEEE Transactions on Computer-Aided Design of Integrated Circuits**, [S.l.], v.19, n.12, p.1523–1543, Dec. 2000.

KORANNE. Formulating SoC Test Scheduling as a Network Transportation Problem. **IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems**, [S.l.], v.21, n.12, p.1517–1525, Dec. 2002.

KORANNE, S.; CHOUDHARY, V. Formulation of SOC Test Scheduling as a Network Transportation Problem. In: DESIGN, AUTOMATION AND TEST IN EUROPE CONFERENCE, 2002, Paris, FR. **Proceedings...** Los Alamitos: IEEE Computer Society, 2002. p.1125.

KORANNE, S.; IYENGAR, V. On the Use of k-tuples for SoC Test Schedule Representation. In: INTERNATIONAL TEST CONFERENCE, 2002. **Proceedings...** Los Alamitos: IEEE Computer Society, 2002. p.539–548.

KUCUKCAKAR, K. Analysis of Emerging Core-based Design Lifecycle. In: IEEE/ACM INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN, 1998. **Digest of Technical Papers...** [S.l.: s.n.], 1998. p.445–449.

KUMAR GOEL, S.; MARINISSEN, E. Layout-driven SOC Test Architecture Design for Test Time and Wire Length Minimization. In: DESIGN, AUTOMATION AND TEST IN EUROPE CONFERENCE, 2003. **Proceedings...** Los Alamitos: IEEE Computer Society, 2003. p.738–743.

LAHIRI, K.; RAGHUNATHAN, A.; DEY, S. Communication Architecture Based Power Management for Battery Efficient System Design. In: ACM/IEEE DESIGN AUTOMATION CONFERENCE, 2002, New Orleans, USA. **Proceedings...** New York: ACM, 2002. p.691–696.

LARSSON, E.; ARVIDSSON, K.; FUJIWARA, H.; PENG, Z. Integrated Test Scheduling, Test Parallelization and TAM Design. In: ASIAN TEST SYMPOSIUM, 11., 2002. **Proceedings...** Los Alamitos: IEEE Computer Society, 2002. p.397–404.

LARSSON, E.; FUJIWARA, H. Power Constrained Preemptive TAM Scheduling. In: IEEE EUROPEAN TEST WORKSHOP, 7., 2002. **Proceedings...** Los Alamitos: IEEE Computer Society, 2002. p.119–126.

LARSSON, E.; PENG, Z. Test Scheduling and Scan-chain Division Under Power Constraint. In: ASIAN TEST SYMPOSIUM, 2001. **Proceedings...** Los Alamitos: IEEE Computer Society, 2001. p.259–264.

LARSSON, E.; PENG, Z. An Integrated System-on-Chip Test Framework. In: DESIGN, AUTOMATION AND TEST IN EUROPE CONFERENCE, 2001, Munich, GE. **Proceedings. . .** Los Alamitos: IEEE Computer Society, 2001. p.138–144.

LARSSON, E.; PENG, Z. An Integrated Framework for the Design and Optimization of SOC Test Solutions. **Journal of Electronic Testing: Theory and Applications**, The Netherlands, v.18, n.4, p.385–400, Aug. 2002.

LARSSON, E.; PENG, Z.; CARLSSON, G. The Design and Optimization of SOC Test Solutions. In: IEEE/ACM INTERNATIONAL CONFERENCE ON COMPUTER AIDED DESIGN, 2001. **Proceedings. . .** [S.l.: s.n.], 2001. p.523–530.

LEE, K.-J.; HUANG, C.-I. A Hierarchical Test Control Architecture for Core Based Design. In: ASIAN TEST SYMPOSIUM, 2000. **Proceedings. . .** Los Alamitos: IEEE Computer Society, 2000. p.248–253.

LI, J.-F.; HUANG, H.-J.; CHEN, J.-B.; SU, C.-P.; WU, C.-W.; CHENG, C.; CHEN, S.-I.; HWANG, C.-Y.; LIN, H.-P. A Hierarchical Test Methodology for Systems on Chip. **IEEE Micro**, [S.l.], v.22, n.5, p.69–81, Sep./Oct. 2002.

LI, J.-F.; HUANG, H.-J.; CHEN, J.-B.; SU, C.-P.; WU, C.-W.; CHENG, C.; CHEN, S.-I.; HWANG, C.-Y.; LIN, H.-P. A Hierarchical Test Scheme for System-on-chip Designs. In: DESIGN, AUTOMATION AND TEST IN EUROPE CONFERENCE, 2002, Paris, FR. **Proceedings. . .** Los Alamitos: IEEE Computer Society, 2002. p.486–490.

LOUSBERG, M. TAPs All Over my Chips. In: INTERNATIONAL TEST CONFERENCE, 2002. **Proceedings. . .** Los Alamitos: IEEE Computer Society, 2002. p.1189.

MAKRIS, Y.; ORAILOGLU, A. RTL Test Justification and Propagation Analysis for Modular Designs. **Journal of Electronic Testing: Theory and Applications**, The Netherlands, v.13, n.2, p.105–120, Oct. 1998.

MARINISSEN, E.; ARENDSEN, R.; BOS, G.; DINGEMANSE, H.; LOUSBERG, M.; WOUTERS, C. A Structured and Scalable Mechanism for Test Access to Embedded Reusable Cores. In: INTERNATIONAL TEST CONFERENCE, 1998. **Proceedings. . .** Los Alamitos: IEEE Computer Society, 1998. p.284–293.

MARINISSEN, E.; GOEL, S.; LOUSBERG, M. Wrapper Design for Embedded Core Test. In: INTERNATIONAL TEST CONFERENCE, 2000. **Proceedings. . .** Los Alamitos: IEEE Computer Society, 2000. p.911–920.

MARINISSEN, E. J.; IYENGAR, V.; CHAKRABARTY, K. A Set of Benchmarks for Modular Testing of SOCs. In: INTERNATIONAL TEST CONFERENCE, 2002. **Proceedings. . .** Los Alamitos: IEEE Computer Society, 2002. p.521–528.

MARINISSEN, E. J.; IYENGAR, V.; CHAKRABARTY, K. **ITC'02 Soc Test Benchmarks**. [S.l.: s.n.], 2003. Available at: <http://www.extra.research.philips.com/itc02socbenchm/>. Acessed in: August 2003.

MARINISSEN, E. J.; KAPUR, R.; LOUSBERG, M.; MCLAURIN, T.; RICCHETTI, M.; ZORIAN, Y. On IEEE P1500's Standard for Embedded Core Test. **Journal of Electronic Testing: Theory and Applications**, The Netherlands, v.18, n.4, p.365–383, Aug. 2002.

MARINISSEN, E. J.; KAPUR, R.; ZORIAN, Y. On Using IEEE P1500 SECT for Test Plug-n-Play. In: INTERNATIONAL TEST CONFERENCE, 2000. **Proceedings...** Los Alamitos: IEEE Computer Society, 2000. p.770–777.

MARINISSEN, E.; ZORIAN, Y. Challenges in Testing Core-based System ICs. **IEEE Communications Magazine**, [S.l.], v.37, n.6, p.104–109, June 1999.

MARINISSEN, E.; ZORIAN, Y.; KAPUR, R.; TAYLOR, T.; WHETSEL, L. Towards a Standard for Embedded Core Test: an Example. In: INTERNATIONAL TEST CONFERENCE, 1999. **Proceedings...** Los Alamitos: IEEE Computer Society, 1999. p.616–627.

MATHWORKS. **MATLAB**: the Language of Technical Computing. [S.l.]: Natick, USA, MathWorks, 1997.

MURESAN, V.; WANG, X.; VLADUTIU, M. A Comparison of Classical Scheduling Approaches in Power-constrained Block-test Scheduling. In: INTERNATIONAL TEST CONFERENCE, 2000. **Proceedings...** Los Alamitos: IEEE Computer Society, 2000. p.882–891.

NAHVI, M.; IVANOV, A. A Packet Switching Communication-based Test Access Mechanism for System Chips. In: IEEE EUROPEAN TEST WORKSHOP, 2001. **Proceedings...** Los Alamitos: IEEE Computer Society, 2001. p.81–86.

NEGREIROS, M.; CARRO, L.; SUSIN, A. Ultra Low Cost Analog BIST Using Spectral Analysis. In: IEEE VLSI TEST SYMPOSIUM, 2003, Napa Valley, USA. **Proceedings...** Los Alamitos: IEEE Computer Society, 2003. p.77–82.

NOURANI, M.; PAPACHRISTOU, C. A Bypass Scheme for Core-based System Fault Testing. In: DESIGN, AUTOMATION AND TEST IN EUROPE CONFERENCE, 1998, Paris, FR. **Proceedings...** Los Alamitos: IEEE Computer Society, 1998. p.979–980.

NOURANI, M.; PAPACHRISTOU, C. Parallelism in Structural Fault Testing of Embedded Cores. In: IEEE VLSI TEST SYMPOSIUM, 16., 1998. **Proceedings...** Los Alamitos: IEEE Computer Society, 1998. p.15–20.

NOURANI, M.; PAPACHRISTOU, C. Structural Fault Testing of Embedded Cores Using Pipelining. **Journal of Electronic Testing: Theory and Applications**, The Netherlands, v.15, n.1-2, p.129–144, Aug./Oct. 1999.

NOURANI, M.; PAPACHRISTOU, C. An ILP Formulation to Optimize Test Access Mechanism in System-on-chip Testing. In: INTERNATIONAL TEST CONFERENCE, 2000. **Proceedings...** Los Alamitos: IEEE Computer Society, 2000. p.902–910.

OAKLAND, S. Considerations for Implementing IEEE 1149.1 on System-on-a-chip Integrated Circuits. In: INTERNATIONAL TEST CONFERENCE, 2000. **Proceedings...** Los Alamitos: IEEE Computer Society, 2000. p.628–637.

PAPACHRISTOU, C. A.; MARTIN, F.; NOURANI, M. Microprocessor Based Testing for Core-Based System on Chip. In: ACM/IEEE DESIGN AUTOMATION CONFERENCE, 1999, New Orleans, USA. **Proceedings...** New York: ACM, 1999. p.586–591.

POMERANZ, I.; REDDY, S. A Partitioning and Storage Based Built-in Test Pattern Generation Method for Delay Faults in Scan Circuits. In: ASIAN TEST SYMPOSIUM, 2002. **Proceedings...** [S.l.: s.n.], 2002. p.110–115.

RAVIKUMAR, C.; VERMA, A.; CHANDRA, G. A Polynomial-time Algorithm for Power Constrained Testing of Core-Based Systems. In: ASIAN TEST SYMPOSIUM, 1999. **Proceedings...** Los Alamitos: IEEE Computer Society Press, 1999. p.107–112.

ROSINGER, P.; AL-HASHIMI, B.; NICOLICI, N. Power Constrained Test Scheduling Using Power Profile Manipulation. In: IEEE INTERNATIONAL SYMPOSIUM ON CIRCUITS AND SYSTEMS, 2001. **Proceedings...** Los Alamitos: IEEE Computer Society, 2001. p.251–254. v. 5.

ROSINGER, P.; GONCIARI, P.; AL-HASHIMI, B.; NICOLICI, N. Simultaneous Reduction in Volume of Test Data and Power Dissipation for Systems-on-a-chip. **Electronic Letters**, [S.l.], v.37, n.24, p.1434–1436, Nov. 2001.

ROSINGER, P. M.; AL-HASHIMI, B. M.; NICOLICI, N. Power Profile Manipulation: A New Approach for Reducing Test Application Time Under Power Constraints. **IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems**, [S.l.], v.21, n.10, p.1217–1225, Oct. 2002.

SANGIOVANNI-VINCENTELLI, A.; MARTIN, G. Platform-Based Design and Software Design Methodology for Embedded Systems. **IEEE Design & Test of Computers**, [S.l.], v.18, n.6, p.23 –33, Nov./Dec. 2001.

SIA. **The National Technology Roadmap for Semiconductors**. [S.l.: s.n.], 1997. Semiconductor Industry Association.

SIA. **The International Technology Roadmap for Semiconductors (ITRS)**. [S.l.: s.n.], 1999. Semiconductor Industry Association.

SINANOGLU, O.; ORAILOGLU, A. Efficient Construction of Aliasing-free Compaction Circuitry. **IEEE Micro**, [S.l.], v.22, n.5, p.82–92, Sep./Oct. 2002.

SKIENA, S. S. **The Algorithm Design Manual**. [S.l.]: New York, Springer-Verlag, 1998.

SUGIHARA, M.; DATE, H.; YASUURA, H. A Novel Test Methodology for Core-based System LSIs and a Testing Time Minimization Problem. In: INTERNATIONAL TEST CONFERENCE, 1998. **Proceedings...** Los Alamitos: IEEE Computer Society, 1998. p.465–472.

SUGIHARA, M.; DATE, H.; YASUURA, H. Analysis and Minimization of Test Time in a Combined BIST and External Test Approach. In: DESIGN, AUTOMATION AND TEST IN EUROPE CONFERENCE, 2000, Paris, FR. **Proceedings...** Los Alamitos: IEEE Computer Society, 2000. p.134–140.

TOUBA. Deterministic Test Vector Compression/Decompression for Systems-on-a-Chip Using an Embedded Processor. **Journal of Electronic Testing: Theory and Applications**, The Netherlands, v.18, n.4, p.503–514, Aug. 2002.

VARMA, P.; BHATIA, S. A Structured Test Re-use Methodology for Core-based System Chips. In: INTERNATIONAL TEST CONFERENCE, 1998. **Proceedings...** Los Alamitos: IEEE Computer Society, 1998. p.294–302.

WHETSEL, L. An IEEE 1149.1 Based Test Access Architecture for ICs with Embedded Cores. In: INTERNATIONAL TEST CONFERENCE, 1997, Washington, USA. **Proceedings...** Los Alamitos: IEEE Computer Society, 1997. p.69–78.

YONEDA, T.; FUJIWARA, H. Design for Consecutive Testability of System-on-a-Chip with Built-In Self Testable Cores. **Journal of Electronic Testing: Theory and Applications**, The Netherlands, v.18, n.4, p.487–501, Aug. 2002.

ZEFERINO, C. A. **SoCIN**:A Parametric and Scalable Network-on-Chip. 2003. Ph.D. Thesis — (Programa de Pós-Graduação em Computação) - Instituto de Informática, UFRGS, Porto Alegre.

ZEFERINO, C.; KREUTZ, M.; CARRO, L.; SUSIN, A. A study on Communication Issues For Systems-on-Chip. In: SYMPOSIUM ON INTEGRATED CIRCUITS AND SYSTEMS DESIGN, 2002, Porto Alegre, Brazil. **Proceedings...** Los Alamitos: IEEE Computer Society, 2002. p.121–126.

ZEFERINO, C.; SUSIN, A. SoCIN:A Parametric and Scalable Network-on-Chip. In: SYMPOSIUM ON INTEGRATED CIRCUITS AND SYSTEMS DESIGN, 2003, São Paulo, Brazil. **Proceedings...** Los Alamitos: IEEE Computer Society, 2003. p.169–174.

ZHAO, D.; UPADHYAYA, S. Adaptive Test Scheduling in SoC's by Dynamic Partitioning. In: IEEE INTERNATIONAL SYMPOSIUM ON DEFECT AND FAULT TOLERANCE IN VLSI SYSTEMS, 17., 2002. **Proceedings...** [S.l.: s.n.], 2002. p.334–342.

ZORIAN, Y. A Distributed BIST Control Scheme for Complex VLSI Devices. In: VLSI TEST SYMPOSIUM, 1993, Princeton, USA. **Proceedings...** Los Alamitos: IEEE Computer Society, 1993. p.6–11.

ZORIAN, Y. Test Requirements for Embedded Core-based Systems and IEEE P1500. In: INTERNATIONAL TEST CONFERENCE, 1997, Washington, USA. **Proceedings...** Los Alamitos: IEEE Computer Society, 1997. p.191–199.

ZORIAN, Y. System-chip Test Strategies. In: ACM/IEEE DESIGN AUTOMATION CONFERENCE, 1998. **Proceedings...** New York: ACM, 1998. p.752–757.

ZORIAN, Y.; DEY, S.; RODGERS, M. Test of Future System-on-chips. In: IEEE/ACM INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN, 2000. **Proceedings...** [S.l.: s.n.], 2000. p.392–398.

ZORIAN, Y.; MARINISSEN, E. J.; DEY, S. Testing Embedded-Core Based System Chips. In: INTERNATIONAL TEST CONFERENCE, 1998, Washington, USA. **Proceedings...** Los Alamitos: IEEE Computer Society, 1998. p.130–143.

ZOU, W.; REDDY, S.; POMERANZ, I.; HUANG, Y. SOC Test Scheduling Using Simulated Annealing. In: IEEE VLSI TEST SYMPOSIUM, 2003, Napa Valley, USA. **Proceedings...** Los Alamitos: IEEE Computer Society, 2003. p.325–330.

APPENDIX A PLANEJAMENTO DE TESTE BASEADO EM REUSO PARA SISTEMAS EM SILÍCIO

O projeto de sistemas eletrônicos atuais segue o paradigma do reuso de componentes de hardware. Neste paradigma, os blocos funcionais, chamados de *componentes virtuais* ou *núcleos de hardware*, são previamente projetados e disponibilizados em algum formato lógico para que sejam acoplados a outros componentes e, posteriormente, sintetizados em um único substrato. Circuitos construídos sob este paradigma são conhecidos como sistemas integrados ou sistemas baseados em componentes virtuais.

Embora esta metodologia de projeto apresente diversas vantagens para a redução da complexidade do projeto de um *chip*, ela criou uma série de novos desafios para o projetista do sistema, não somente em relação ao projeto, mas também em relação ao teste do produto final. Na verdade, o teste dos sistemas integrados tem se tornado uma das tarefas mais custosas no processo de fabricação do chip (ZORIAN; DEY; RODGERS, 2000). O acesso aos núcleos profundamente embutidos no sistema, a integração dos diversos métodos de teste e a otimização dos diversos fatores de custo do sistema, como área, tempo de teste e número de pinos no encapsulamento, são apenas alguns dos problemas que precisam ser resolvidos durante o planejamento do teste de produção do novo circuito. Além disso, a mudança no fluxo de informações sobre o sistema é outro fator complicante para a definição de um mecanismo de teste eficaz, já que os projetistas dos núcleos de hardware não são, necessariamente, os projetistas do sistema. Neste contexto, esta tese propõe duas abordagens para o planejamento de teste de sistemas integrados. As abordagens propostas têm como principal objetivo a redução dos custos de teste do sistema através do reuso dos recursos de hardware disponíveis no sistema sob teste e da integração do planejamento de teste no fluxo de projeto do circuito.

Neste texto, os principais problemas relacionados ao teste dos sistemas integrados baseados em componentes virtuais são, inicialmente, identificados no Capítulo 2. Estes problemas podem ser divididos em três grupos: requisitos de teste dos componentes virtuais, requisitos de teste de interconexões e requisitos de teste do sistema.

Em relação aos componentes virtuais, os principais requisitos são:

- definição do teste de cada núcleo embutido no sistema;
- definição do mecanismo de acesso a cada núcleo durante o teste;
- definição de um mecanismo de isolamento do núcleo em relação ao sistema durante o teste;

O teste de interconexões tem um único requisito: a possibilidade de controlar e observar cada conexão. Contudo, esta parte do teste é de extrema importância para a caracteri-

zação do sistema, pois ela determina o real desempenho conseguido pelo sistema.

Os requisitos do sistema como um todo sao:

- teste da lógica definida pelo integrador do sistema;
- definição do escalonamento de teste;
- definição de um controlador de teste
- integração dos mecanismos de teste de cada núcleo

No Capítulo 3, são discutidas as principais soluções apresentadas na literatura para os problemas listados acima.

Até o ano 2000, quando esta tese foi definida, a maior parte das soluções existentes tratava da definição do mecanismo de acesso aos núcleos embutidos no sistema. Estes mecanismos eram definidos considerando um sistema completamente definido, ou seja, o planejamento de teste era feito em uma etapa final do projeto, quando se torna mais difícil modificar decisões de projeto. As poucas abordagens que permitiam o planejamento de teste no início do projeto do sistema, otimizavam apenas um sub-conjunto dos custos de teste ou tornavam a solução muito dependente da experiência do projetista. Em geral, os métodos de teste propostos assumiam modelos restritos de mecanismos de acesso (apenas barramentos ou apenas conexões funcionais) e, a partir deste modelo restrito, otimizavam os outros custos, como tempo de teste e área. Assim, cabia ao integrador do sistema fazer a exploração do espaço de projeto considerando os custos de teste, fornecendo, a cada passo, diferentes recursos de teste como pinos, controladores BIST, etc. As decisões de teste eram, então, mais baseadas na experiência do projetista do que nas características de cada projeto.

Porém, o projeto de um sistema lida com uma série de restrições e compromisos entre os diferentes fatores de custo, como área, desempenho e potência, por exemplo. A otimização dos fatores de custo do teste, como acréscimo de área, pinos e potência, não pode ser feita de forma isolada. O projetista do sistema precisa combinar e equilibrar os custos de teste com os custos de projeto em si. Assim, decisões de projeto podem interferir nos custos de teste e, vice-versa, decisões sobre o teste podem influenciar os custos de projeto. Por exemplo, o acréscimo de área acarretado pela inclusão de estruturas de teste dentro do chip implica aumento de área e, como conseqüência, redução do *yield*. Porém, o integrador do sistema não é, em geral, um especialista em teste e torna-se difícil prever o impacto de uma decisão de projeto nos custos de teste finais. Da mesma forma, o especialista em teste nem sempre está diretamente envolvido nas decisões de projeto ou mesmo na integração e síntese do sistema. O projetista procura uma solução que represente o melhor compromisso de custos que atenda às especificações da aplicação. O engenheiro de teste, por outro lado, está mais preocupado com a otimização de um ou dois fatores de custo principais, como o tempo de teste e o número de pinos extras na interface do sistema. As decisões deste último, portanto, nem sempre são as mais adequadas do ponto de vista do projeto. Assim, a inclusão do planejamento de teste no início do ciclo de projeto do sistema é a chave para a redução da complexidade e do custo de teste de sistemas complexos.

Neste contexto, definiu-se como principal objetivo desta tese trazer o planejamento do teste de um sistema para o início do projeto, permitindo a exploração do espaço de projeto levando-se em conta os custos de teste. Este objetivo foi atingido através da definição de duas abordagens de teste, definidas de acordo com o modelo de conexão do sistema.

Foram definidas duas ferramentas de auxílio ao projetista que estimam os custos de teste a partir das características de cada sistema. Assim, o projetista pode ter uma estimativa do custo de uma decisão de projeto antes mesmo da síntese, podendo combinar os requisitos do projeto com os requisitos de teste. Por outro lado, o projetista do núcleo também pode fazer uso das ferramentas propostas para estimar o impacto do componente em um futuro sistema e definir um núcleo que seja mais facilmente testável, por exemplo.

A primeira abordagem proposta neste trabalho é detalhada no Capítulo 4 e considera os sistemas cujos componentes se comunicam através de conexões dedicadas ou barramentos funcionais. Para estes sistemas, o planejamento do teste consiste na definição de um mecanismo de acesso aos componentes internos do circuito e de um mecanismo para exploração do espaço de projeto. O mecanismo de acesso prevê o reuso das conexões funcionais assim como o uso de barramentos de teste locais, núcleos transparentes e outros modos de passagem do sinal de teste implementado no núcleo ou no seu *wrapper*. O algoritmo de escalonamento de teste é definido juntamente com o mecanismo de acesso de forma que diferentes combinações de custos em pinos, área e tempo de teste sejam explorados. Além disso, restrições de consumo de potência do sistema podem ser consideradas durante o escalonamento dos testes. Esta expansão dos conceitos de mecanismo de acesso e dos parâmetros de otimização possibilita uma pesquisa detalhada, mas eficiente, no imenso espaço de projeto destes sistemas. Os resultados experimentais apresentados para este método mostram claramente a variedade de soluções que podem ser exploradas e a eficiência desta abordagem na otimização do teste de um sistema complexo.

As principais contribuições desta primeira abordagem, batizada de ReBaTe (**R**euse-**B**ased **T**est **P**lanning), em relação às soluções existentes na literatura são:

1. não se assume um único tipo de mecanismo de acesso para o sistema. Barramentos de teste são considerados juntamente com o reuso de conexões funcionais, modos de passagem (transparência) eventualmente implementados por núcleos, e modos de passagem (*bypass*) implementados pelos *wrappers*;

2. a solução não otimiza totalmente cada núcleo de forma a minimizar o custo de teste do sistema. Ao contrário, a diversidade dos requisitos de teste dos núcleos é explorada de forma que núcleos que possuam mais requisitos de teste sejam privilegiados com mais recursos (pinos, tempo, etc), enquanto núcleos menos críticos usam menos recursos ou recursos mais caros que não representam um grande impacto no custo do sistema como um todo;

3. o escalonamento de teste e o mecanismo de acesso a cada núcleo são definidos em paralelo e não como tarefas independentes, como ocorre em outras soluções onde o escalonamento é definido a partir de um mecanismo de acesso fixo. Este é um dos aspectos que permite a exploração do espaço de projeto de forma que o melhor compromisso entre os custos de tempo de teste, área e pinos seja definido.

Diferentes aspectos desta técnica são apresentados em três artigos: (COTA et al., 2002), (COTA et al., 2002) e (COTA et al., 2003).

Redes em-chip serão, provavelmente, a principal plataforma de comunicação dos sistemas integrados, substituindo os atuais barramentos. Assim, a segunda abordagem de planejamento de teste apresentada nesta tese propõe o reuso da rede em-chip como mecanismo de acesso aos componentes internos aos sistemas que usam esta plataforma de comunicação. Um algoritmo de escalonamento de teste que considera as restrições de potência

da aplicação é apresentado e a estratégia de teste é avaliada para diferentes configurações do sistema, como o número de interfaces com o testador, diferentes posicionamentos dos núcleos nos roteadores da rede e diferentes restrições para o consumo de potência. Os resultados experimentais mostram que a capacidade de paralelização da rede em-chip pode ser explorada para reduzir o tempo de teste do sistema, enquanto os custos de área e pinos de teste são drasticamente minimizados. Esta é a primeira abordagem a considerar o reuso da rede em-chip como mecanismo de acesso durante o teste e está sendo chamada de NoCBaTe (**NoC-B**ased **T**esting). Este método é apresentado no *VLSI Test Symposium (VTS)* (COTA et al., 2003), no *European Test Workshop (ETW)* (COTA et al., 2003a) e no *International Test Conference (ITC)* (COTA et al., 2003b) de 2003.

Ambas as técnicas são validadas através dos sistemas disponíveis no *ITC'02 SoC Test Benchmarks*. Dentre estes sistemas, o benchmark u226 é uma contribuição desta Universidade, resultado de um exemplo utilizado no artigo (COTA et al., 2002) que primeiro apresentou a técnica ReBaTe.

No Capítulo 6, as duas técnicas são comparadas entre si e com outras técnicas de teste apresentadas recentemente. A partir das comparações das técnicas propostas, pode-se concluir que o teste baseado em redes em-chip apresenta um bom compromisso entre custo e tempo de teste principalmente para os sistemas com grande número de núcleos e que tenham um maior número de interfaces que podem ser reusadas durante o teste. Para sistemas com pequeno numero de núcleos de hardware, a rede em-chip apresenta um pequeno número de roteadores, e, conseqüentemente, um pequeno número de caminhos de acesso que podem ser usados ao mesmo tempo. Neste caso, o método baseado no reuso das conexões funcionais apresenta melhores tempos de teste a um custo geralmente baixo em termos de pinos extras na interface do sistema e acréscimo de área.

A comparação deste trabalho com outras técnicas de teste apresentadas recentemente, por outro lado, confirma a eficácia dos métodos propostos e pode ser vista como mais uma contribuição deste trabalho, uma vez que não foi encontrado, na literatura atual, nenhum trabalho comparativo entre métodos baseados em barramentos de teste e métodos baseados no reuso de conexões funcionais. Por fim, o uso dos benchmarks de sistemas integrados presentes no conjunto ITC'02, permitiu a definição de algumas métricas e regras que devem ser observadas na avaliação de métodos que lidam com o sistema de maneira geral e não apenas com as informações dos núcleos embutidos. Como os benchmarks provêem informações apenas sobre os núcleos e seus requisitos de teste, diferentes suposições sobre os dados do sistema como conexões entre núcleos, localização dos núcleos no *chip*, etc. levam a diferentes resultados em termos de custos de teste. Esta discussão foi apresentada no *Latin-American Test Workshop (LATW)* de 2003 (COTA; CARRO; LUBASZEWSKI, 2003a);

De modo geral, a principal contribuição técnica deste trabalho é a expansão dos conceitos de mecanismo de acesso de teste e planejamento de teste para sistemas integrados de hardware. A maior parte dos trabalhos nesta área considera apenas o barramento de teste como forma de acesso e procuram minimizar os custos de definição e implementação deste mecanismo, bem como o tempo de teste resultante para o sistema, sem considerar aspectos do sistema como um todo. Nesta tese, o espaço de projeto do sistema é explorado levando-se em conta os custos de teste e, vice-versa, o teste é definido de acordo com as características e restrições de cada projeto. Assim, por exemplo, o projetista pode verificar, antes da síntese, o impacto do modelo de conexão do sistema, dos núcleos escolhidos e até mesmo da localização dos núcleos nos custos de teste do *chip*.

Embora não apresentado neste texto, a definição de um controlador de teste embutido

no sistema é um trabalho derivado deste estudo e foi desenvolvido como projeto de disciplina e, posteriormente, como uma dissertação de mestrado. O controlador MET, ou **M***icroprocessor for* **E***mbedded* **T***est* foi inicialmente pensado para o teste *at-speed* de núcleos profundamente embutidos no sistema. Sua primeira versão foi apresentada no *5th IEEE Workshop on Testing Embedded Core-based Systems* (COTA et al., 2001). Estudos para avaliação do desempenho e requisitos de memória para uso do controlador foram realizados e apresentados em (CASSOL, 2002).

Apesar dos resultados positivos apresentados para as técnicas desenvolvidas neste trabalho, há, ainda, alguns pontos que podem ser aprimorados:

1. o estudo de outras heurísticas para estabelecer o espaço de pesquisa da ferramenta ReBaTe;

2. a inclusão do controlador MET como um recurso de teste na ferramenta ReBaTe, de forma que teste *at-speed* possa ser tratado;

3. a otimização dos *wrappers* antes da definição do mecanismo de acesso (método ReBaTe) e dos pacotes de teste (método NoCBate). A otimização proposta em (IYENGAR; CHAKRABARTY; MARINISSEN, 2001), por exemplo, define um conjunto de cadeias de varredura no wrapper de forma a minimizar o tempo de teste do núcleo antes de se otimizar os custos do sistema;

4. a implementação em VHDL de um dos benchmarks disponíveis, para validação das estimativas de área, tempo de teste e configurações assumidas nas duas técnicas apresentadas;

5. o estudo de um método para geração automática de *wrappers* que implementem não só as regras estabelecidas pelo padrão P1500, mas também os modos de operação necessários para as soluções definidas nos métodos propostos. Este estudo deve verificar, ainda, os requisitos em termos de sinais de controle e configuração necessários para os *wrappers*;

6. a consideração de outras topologias de rede e outros algoritmos de roteamento (roteamento dedicado, por exemplo) na técnica NoCBate;

7. no método NoCBaTe, a verificação do real impacto das interfaces funcionais de largura menor que a largura do canal no tempo de teste do sistema;

8. a definição de outros modelos de pacotes de teste para o método baseado na rede em-chip e verificação do tempo de teste resultante e do acréscimo de área para modificação dos *wrappers*;

9. embora os resultados apresentados neste texto tenham sido gerados considerando um modelo de consumo de potência realista, este modelo é estimado e teórico. Assim, o refinamento deste modelo é um ponto importante. Por exemplo, pode-se considerar canais de comunicação na rede em-chip de tamanhos diferentes ou modelos de consumo mais detalhados para os núcleos, como o modelo proposto em (ROSINGER; AL-HASHIMI; NICOLICI, 2002).

Finalmente, considerando o teste de sistemas integrados de forma mais abrangente, os seguintes temas de pesquisa ainda apresentam pontos em aberto:

- a inclusão do teste de blocos analógicos no planejamento de teste do sistema, continuando a linha de exploração das cracterísticas gerais de cada projeto. Núcleos analógicos auto-testáveis, por exemplo, podem apresentar tempos de teste maiores que os blocos digitais e este tempo pode ser usado para reduzir os custos dos mecanismos de acesso digitais. O benchmark u226 é um exemplo de um sistema que apresenta esta característica. Além disso, técnicas de teste para núcleos analógicos baseados no reuso de processadores digitais presentes no sistema integrado têm sido estudadas neste grupo de pesquisa (NEGREIROS; CARRO; SUSIN, 2003). Assim, o escalonamento de teste para o sistema como um todo deve levar em consideração este fato de forma que o teste seja consistente;

- núcleos auto-testáveis são considerados uma poderosa solução para reduzir os requisitos de acesso do sistema sobre o testador externo. Contudo, o impacto de tais blocos no planejamento de teste ainda não foi extensivamente estudado. Resultados preliminares considerando a definição de um conjunto de núcleos auto-testáveis necessário e suficiente para minimização do tempo de teste do sistema foi apresentado no *Test Resource Partitioning (TRP) Workshop* de 2003 (COTA; CARRO; LUBASZEWSKI, 2003b). Porém, um estudo mais detalhado do método proposto é necessário, bem como sua inclusão nas ferramentas de planejamento de teste desenvolvidas nesta tese.

APPENDIX B CD-ROM DESCRIPTION

The programs that implement the two proposed test planning methods, and the respective results for the ITC'02 benchmarks are available in the CD-ROM that accompanies this manuscript.

The CD-ROM contains the following directory structure:

- **NoCBaTe:** This directory contains the source, executable, and example files of the NoCBaTe tool, version 3.1.
  - **NoCBaTe_benchs:** Description of ITC'02 SoC Test Benchmarks in the format required by the NoCBaTe tool. For each benchmark, there is a Matlab file describing the system. This Matlab file is called by the function "gera_inicio", which generates the input files .soc and .noc for the tool. For the current set of benchmarks, those files are already available.
  - **NoCBaTe_v3.1:** Source and executable of the NoCBaTe tool. To run the scheduling, execute the file "scheduling2.exe". Three files are required as input: a _in.soc, describing the system, a _in.noc, describing the network-on-chip, and a _topology.sav, describing the placement of the cores into the network. All files must have the same name as prefix.
- **Placement:** This directory contains the placement tools locally developed for the placement of cores considering both, the core-to-core connection model, and the NoC-based connection. The tools are built using a Simulated Annealing heuristic.
  - **Core-to-core:** Placement tool for cores directly connected in a SoC. An ASCII file describing the functional connections among cores, and the dimensions of each core is used as input. As output, a table of distances among cores is generated and saved into a text file.
  - **NoC_based:** Placement tool for cores connected using a NoC platform. An ASCII file describing the communication requirements of the system is used as input. this file also contains the network dimensions (number of rows and columns). The output of the placement is a text file indicating the location of each core.
- **ReBaTe:** This directory contains the ReBaTe tool, version 2.4.
  - **ReBaTe_benchs:** Description of ITC'02 SoC Test Benchmarks in the format required by the ReBaTe tool. For each benchmark, there is a Matlab file

describing the system. This Matlab file is called by the function "scheduling" present in the directory *ReBaTe_v2.4*. Additionally, the results presented in this manuscript are also present in this directory, as well as the resulting placement for estimated functional connections.

  - **ReBaTe_v2.4:** Source and executable of the ReBaTe tool. To run the test planning, execute the file "scheduling" in the Matlab command window. The Matlab file describing the system (as exemplified for the current set of benchmarks), is required as input and must be informed in the beginning of the file *scheduling.m*.

- **Thesis**
  - **Presentation:** contains the Powerpoint file with the final presentation of this thesis.
  - **Text:** LaTeX files and figures used to generate this manuscript

- **Publications:** PDF files of the author's publications related to this thesis. The complete reference of each paper is described in the file *ErikaCota_publications.pdf*.