IEEE Copyright Notice

Copyright © 2002 IEEE.
Reprinted from Proceedings of IEEE International Test Conference 2002.

This material is posted here with permission of the IEEE. Internal or personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution must be obtained from the IEEE by sending an e-mail message to pubs-permissions@ieee.org.

By choosing to view this document, you agree to all provisions of the copyright laws protecting it.

<u>Note:</u> *The original ITC'02 paper referred by mistake to benchmark SOC p22810 as 'p22081'. In this version of the paper, that mistake is corrected.*

Paper 19.1 at
IEEE International Test Conference (ITC'02)
Baltimore, MD, USA – October 2002 (NL-R 21.832)

A Set of Benchmarks for Modular Testing of SOCs

Erik Jan Marinissen

Philips Research Laboratories
IC Design – Digital Design & Test
Prof. Holstlaan 4, M/S WAY-41
5656 AA Eindhoven, The Netherlands
erik.jan.marinissen@philips.com

Vikram Iyengar*

IBM Microelectronics
ASIC Test and Methodology Development
1000 River Road
Essex Jct, VT 05452, U.S.A.
vikrami@us.ibm.com

Krishnendu Chakrabarty

Duke University
Dept. of Electrical and Computer Eng.
110 Hudson Hall, Box 90291
Durham, NC 27708-0291, U.S.A.
krish@ee.duke.edu

Abstract

This paper presents the *ITC'02 SOC Test Benchmarks*. The purpose of this new benchmark set 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 paper defines the benchmark format and naming scheme, and presents the benchmark SOCs. In addition, it provides an overview of the research problems that can be addressed and evaluated by means of this benchmark set. These research problems include the design of optimized test access infrastructures and test schedules.

1 Introduction

Integrated circuits (ICs) continue to grow in size and complexity. Today, it is possible to integrate the functionality of a complete system onto one single die. Such an IC is often referred to as a *system-on-chip* (SOC). The test development process for these 'monster chips' is challenging [1], both due to the fact that SOCs typically contain a very heterogeneous mix of circuit structures and design styles, as well as due to the sheer size of SOCs. Modular test development is therefore an attractive proposition for SOCs [2].

SOCs are increasingly designed by embedding pre-designed and pre-verified modules, or *cores*. These embedded cores facilitate the import of specialized design expertise and reduce the SOC design time. Many embedded cores have a non-logic circuit structure; examples are embedded memories, embedded analog modules, and embedded FPGAs. These circuit structures exhibit different defect behavior, and hence demand dedicated tests. Furthermore, for many (logic) cores, the core provider does not reveal the implementation of the core, in order to protect the intellectual property of the design. In those cases, the system integrator has insufficient information about the core's implementation to creat high-quality tests, and hence is forced to apply the tests as provided with the core.

However, also for logic modules whose implementation is known, it pays off to adopt a modular test development process, instead of creating the test patterns in one SOC top-level test generation tool run. Modular test development enables a 'divide-and-conquer' test generation approach, which contains the hard-to-test parts of the SOC to one or a few modules only. Typically, this has positive effects on both the test generation run times, as well as on the test data volume. This effect is multiplied when (last-minute) design changes force test generation iterations. Furthermore, a modular test approach enables test reuse. This especially pays off if a module is used in multiple SOCs, or if subsequent SOC designs are relatively similar (the 'family' concept); both situations are actually quite common.

Modular testing requires test access to the module-under-test. Typically, these modules are deeply embedded in the SOC, and due to their surrounding circuitry, direct access from the SOC pins to the module terminals is not possible. In order to be able to test an embedded module as a stand-alone unit, it should be isolated from its surrounding circuitry and electrical test access needs to be provided. Among the research challenges in SOC testing are the (automated) design of a test access infrastructure and corresponding test schedule that meet all test requirements, and minimizes test application time, required ATE resources, silicon area, power dissipation during test, etc.

Early publications in this emerging research domain have so far suffered from the lack of a common set of benchmark SOCs. Many academic researchers do not have access to realistic data regarding such circuits. Only a few researchers have obtained industrial data to work with, but often that data was provided on an exclusive basis, which does not allow the objective comparison of different competing approaches and techniques.

In this paper we present a new set of benchmarks that aims at addressing the needs mentioned above. Named the *ITC'02 SOC Test Benchmarks*, the set of SOC benchmarks is intended to stimulate research into new methods and tools for modular testing of core-based SOCs and to enable the objective comparison of such methods and tools with respect to

* Part of this work was performed when Vikram Iyengar was with Duke University in Durham, NC, U.S.A.

effectiveness and efficiency. This paper describes the SOC benchmarks and provides an overview of the type of research questions that can be addressed and evaluated by means of this benchmark set.

While the set of benchmarks presented in this paper addresses a pressing need in the SOC test research community, it is important to recognize that their use is limited in scope to problems that involve modular testing of SOCs. These benchmarks are intended *neither* for the evaluation of methods that require the internal details of the SOC implementation, such as fault modeling, fault diagnosis, and test generation using merged cores, *nor* for the evaluation of methods that require the detailed values of the test pattern sets, such as test data compression techniques. Nevertheless, this limitation does not undermine the value of these benchmarks in any way. Modular core-based SOC testing is becoming increasingly popular, and there has been a surge in the number of research papers published on this topic in the past few years [3]. The ITC'02 SOC Test Benchmarks will therefore be especially welcomed by this growing research community.

The sequel of this paper is organized as follows. Section 2 briefly reviews previous benchmarking initiatives. In Section 3, we describe the format of our benchmark data and illustrate this with an example. Section 4 explains the naming convention for our benchmark SOCs. Section 5 presents the set of benchmark SOCs and lists several of their key features. In Section 6, we present a classification of problems and related prior work in SOC test automation, for which these benchmark SOCs can be useful. Section 7 concludes this paper.

2 Prior Work in Benchmarks

Benchmark circuits have been used for many years to allow objective comparison of methods and tools. In the domain of testing integrated circuits, the ISCAS'85 and ISCAS'89 benchmark sets are probably the best-known and most-used sets of benchmarks. The ISCAS'85 benchmarks were first presented by Brglez and Fujiwara [4]. They consist of 10 combinational digital circuits in gate-level netlist representation. The benchmarks can be downloaded through the Internet at [5]. The ISCAS'89 benchmarks were first presented by Brglez, Bryan, and Kozminski [6]. They consist of 31 sequential digital circuits in gate-level netlist representation. The benchmarks can be downloaded through the Internet at [7]. As the full gate-level implementation of these benchmark circuits is provided, they have been used for a wide range of research problems, including automatic test pattern generation, fault simulation, and test data compression. The homogeneous character of the combined ISCAS'85 and ISCAS'89 benchmarks makes them very suited for comparing methods or tools for a list of circuits. The main drawback of the ISCAS benchmarks is that they are outdated. Although realistic in size and complexity at the time they were released, the ISCAS circuits do not represent today's multi-million gate SOC designs. Modern SOCs are designed at register-transfer level (RTL), contain embedded modules such as memories and analog modules, have multiple clock domains, and bidirectional signals and tri-state buses, and none of this exists in the ISCAS benchmarks.

The ITC'99 Benchmarks initiative by Davidson [8, 9] was an attempt to create a new set of benchmarks for the test research community that would overcome the abovementioned drawbacks of the ISCAS benchmarks. The result was a rather heterogeneous and unwieldy set of around thirty benchmarks. Some are from academic sources, others are from industrial sources; a few are combinational, most are sequential; some are gate-level netlists, others RTL source code; some include embedded memories, others do not; some are freely available, others require signing a so-called Community Source License. The benchmarks can be downloaded from the Internet at [10].

Many more sets of benchmark circuits are available through the web site of the Collaborative Benchmarking Laboratory at North Carolina State University: http://www.cbl.ncsu.edu/.

In the benchmark initiative presented in this paper, we tried to combine the strengths of both the ISCAS and ITC'99 benchmark sets. On the one hand, we strive for a homogeneous set of circuits. The type of information provided for the SOCs should be the same, so that if one SOC can be used to obtain experimental results in a particular research project, all other SOCs can be used as well. In this way, research results can be presented on a set of SOCs, hence giving more statistical validity to the conclusions based on the experimental results. On the other hand, we want circuits that are representative for today's SOC designs. An additional goal is to restrict the amount of intellectual property released through the benchmark data, such that companies can relatively easily contribute to the benchmark set the data of one or more of their real-life SOC designs. The information released should of course be such that useful research problems can still be addressed by means of these circuits. In the next section we detail the exact benchmark format, which shows what information is contained in our benchmark data.

3 Benchmark Format

The benchmark format contains the following information per SOC.

- The SOC name according to the naming convention in Section 4.
- The total number of modules in the SOC.
- Global settings that specify whether or not the optional data for layout position and power dissipation are provided.
- Per module in the SOC:
  - The level in the design hierarchy.
  - The number of input, output, and bidirectional terminals.
  - The number of scan chains and their lengths.
  - The absolute layout location of the core (optional).
  - The total number of tests.
  - Per test:
    - Whether or not this test uses the module-internal scan chains and/or the core-external Test Access Mechanism (TAM).
    - The number of test patterns.
    - The power dissipation (optional).

This section describes the benchmark format in detail. First, we explain how a multi-level design hierarchy is expressed in the benchmark format. Subsequently, we give a short description of all keywords used in the benchmark format. Finally, we illustrate the format by means of an example SOC.

3.1 SOC Design Hierarchy

Our SOC benchmark format supports the description of multiple levels of design hierarchy. We can describe the embedding of cores within cores. The format also supports SOC design in which only some pre-designed cores are used, and the remaining SOC circuitry is interconnect wiring, glue logic, and/or user-defined blocks.

A uniform description of the SOC hierarchy is enabled by using the term 'Module' to denote each SOC component that is tested as a stand-alone unit. The SOC itself, i.e., the top-level design unit, is a module. It contains the non-core circuitry. Other modules are embedded in this top-level model. These modules can be pre-designed embedded cores, but also user-defined modules, of which the circuit structure and/or size justify stand-alone testing. And these modules can in their turn contain again other modules, etc.

In order to express the design hierarchy in the benchmark format, we use the keywords `Module` and `Level`. The top-level SOC design is named 'Module 0'. The other modules get assigned consecutive module numbers, based on a depth-first traversal of the design hierarchy tree. The levels correspond with the depth in the design hierarchy tree. 'Module 0' has by definition 'Level 0', whereas a module embedded in a module of 'Level n' has 'Level $n + 1$'.

Figure 1(a) shows an example modular SOC design, containing seven modules. The corresponding design hierarchy tree, consisting of four levels, is shown in Figure 1(b). The corresponding 'Level' and 'Module' names are added in the figure.

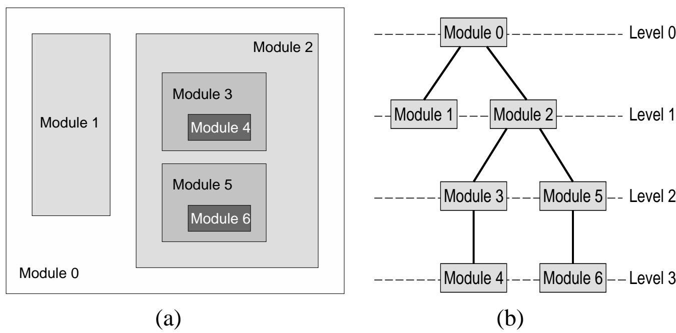

Figure 1: Example modular SOC design (a) and the corresponding module hierarchy tree (b).

3.2 Keywords

The benchmark format uses keywords with the following meaning.

- `SocName`: Uniquely identifying name of the SOC, determined from the description in Section 4.
- `TotalModules`: Total number of modules in this SOC. (A module is either the SOC itself or an embedded core.)
- `Options`: The benchmark format contains both mandatory and optional data. The optional data is `XY` and `Power`. If for a certain type of optional data the value '0' is listed here, this means that none of the modules of the SOC has this optional data provided. In case the value '1' is listed, it means that for some or all modules, this data is specified.

For each module, the following information is specified.

- `Module`: Module number. The SOC itself is `Module 0`. Subsequent modules in the depth-first traversal of the design hierarchy tree get assigned subsequent module numbers.
- `Level`: The level of this module in the SOC design hierarchy. `Level 0` refers to the top of the design hierarchy. If a Module m has `Level n` (for $n > 0$), this means that m is embedded in the module of level $n - 1$ with the largest module number smaller than m.
- `Inputs`, `Outputs`, `Bidirs`: The numbers of input-only, output-only, and bidirectional terminals of the module.
- `ScanChains N :` l_1 l_2 $\cdots$ l_N: The number N of scan chains of this module, and the lengths l_i, in scan flip flops, of these N scan chains. In case the module has no scan chains, the format is: `ScanChains 0 :`.
- `X`, `Y`: The X, Y-coordinate of the center of a core on the SOC layout. X, Y-coordinates are expressed as non-negative integers. The unit and the $(0, 0)$ origin of the coordinates are not specified; however, they should be the same for all modules throughout one SOC benchmark. The fact that all X, Y-coordinates should be non-negative integers, means that the $(0, 0)$ origin cannot be located inside the SOC die itself. For modules in a design hierarchy, the absolute X, Y-coordinates on the SOC layout are mentioned, not the relative ones. This information is optional. It can be omitted for individual or for all modules of the SOC. In case X, Y-coordinates are not specified for an individual core, they are listed as '-1'.
- `TotalTests`: The number of separate tests for the module.

Per test, the following is listed.

- `Test`: The test number for this module. Numbering is consecutive, starts at 1, and ends at `TotalTests`.
- `ScanUse`: This is a binary variable (0/1), that specifies whether or not this test uses the module-internal scan chains. If '0', the module-internal scan chains are *not* used; if '1', the module-internal scan chains are used. It is mandatory to specify this parameter (even though it has no meaning in case the module has no internal scan chains). The setting of this parameter typically influences the number of clock cycles needed to apply one test pattern.
- `TamUse`: This is a binary variable (0/1), that specifies whether or not this test uses a core-external Test Access Mechanism (TAM) [2]. If '0', such a TAM is not used; if '1', a TAM is used. It is mandatory to specify this parameter. For tests for which the stimuli are generated outside the module-under-test and/or the responses are observed outside the module-under-test, a TAM is utilized during the test for transportation of stimuli and responses. This is typically the case for external testing (from an ATE), and also for BIST (Built-In Self Test) implementations for which the BIST engine resides outside the module-under-test. For BIST implementations for which the BIST engine resides inside the core, typically an external TAM is not utilized during the execution of the BIST. In such cases, the TAM bandwidth might be utilized to test another core at the same time.

- `Patterns`: The number of test patterns for this test. One test pattern is defined as one coherent set of stimuli and corresponding responses. For scan testing, one test pattern includes a full load, apply, and unload of the scan chains involved.

- `Power`: Power dissipation per test. Power is expressed as a non-negative integer. The nature (average, peak, otherwise) and unit of power are not specified, but should be the same for all modules throughout one SOC benchmark. This information is optional. It can be omitted for individual or for all modules of the SOC. In case power values are not specified for an individual core test, they are listed as '-1'.

3.3 Example

Figure 2 illustrates the benchmark format by means of an example SOC, named x847.

SOC x847 in Figure 2 has 35 lines with text and some blank lines. Please note that the line numbers are not part of the format, but are added here for explanation purposes only. The benchmark format allows to insert blank lines; typically, this will be used to distinguish between the various module descriptions.

The first three lines provide general information of this SOC. Line 01 says that the SOC name is 'x847'. Line 02 specifies that the total num-

Example 1 [x847.soc]

```
01  SocName x847
02  TotalModules 7
03  Options Power 1 XY 1

04  Module 0 Level 0 Inputs 312 Outputs 312 Bidirs 0 ScanChains 2 :  54 43
05  Module 0 X -1 Y -1
06  Module 0 TotalTests 2
07  Module 0 Test 1 ScanUse 1 TamUse 0 Patterns 43 Power 128
08  Module 0 Test 2 ScanUse 0 TamUse 0 Patterns 32 Power 537

09  Module 1 Level 1 Inputs 10 Outputs 11 Bidirs 12 ScanChains 4 :  20 21 22 23
10  Module 1 X 678 Y 123
11  Module 1 TotalTests 3
12  Module 1 Test 1 ScanUse 1 TamUse 1 Patterns 567 Power 576
13  Module 1 Test 2 ScanUse 1 TamUse 1 Patterns 876 Power 275
14  Module 1 Test 3 ScanUse 0 TamUse 1 Patterns 908 Power 123

15  Module 2 Level 1 Inputs 44 Outputs 46 Bidirs 0 ScanChains 1 :  100
16  Module 2 X 324 Y 98
17  Module 2 TotalTests 2
18  Module 2 Test 1 ScanUse 1 TamUse 1 Patterns 4356 Power 1334
19  Module 2 Test 2 ScanUse 1 TamUse 1 Patterns 56 Power 2245

20  Module 3 Level 2 Inputs 312 Outputs 312 Bidirs 0 ScanChains 2 :  75 75
21  Module 3 X 304 Y 80
22  Module 3 TotalTests 1
23  Module 3 Test 1 ScanUse 1 TamUse 1 Patterns 25 Power -1

24  Module 4 Level 3 Inputs 112 Outputs 543 Bidirs 23 ScanChains 0 :
25  Module 4 X -1 Y -1
26  Module 4 TotalTests 1
27  Module 4 Test 1 ScanUse 1 TamUse 0 Patterns 12 Power -1

28  Module 5 Level 2 Inputs 312 Outputs 312 Bidirs 0 ScanChains 2 :  75 75
29  Module 5 X 344 Y 80
30  Module 5 TotalTests 1
31  Module 5 Test 1 ScanUse 1 TamUse 1 Patterns 25 Power -1

32  Module 6 Level 3 Inputs 112 Outputs 543 Bidirs 23 ScanChains 0 :
33  Module 6 X -1 Y -1
34  Module 6 TotalTests 1
35  Module 6 Test 1 ScanUse 1 TamUse 0 Patterns 12 Power -1
```

Figure 2: Example benchmark SOC x847.

ber of modules in this SOC is seven. `Options` in Line 03 lists whether power and layout co-ordinate values for modules are supplied for this benchmark SOC. Both options are listed as '1', which means that both X, Y layout coordinates and power dissipation values are specified for one or more modules.

The next 33 lines describe the seven modules in the SOC. Lines 04–08 describe Module 0. Module 0 has `Level 0`, which means that Module 0 is the top-level module. This is in fact always the case for Module 0. Module 0 has 312 inputs, 312 outputs, and 0 bidirectional terminals; as Module 0 is the top-level SOC design, these terminals directly correspond to SOC pins. Module 0 contains two scan chains, of lengths 54 and 43 scan flip flops, respectively. Line 05 shows that no X, Y layout coordinates are specified for Module 0. Module 0 has two tests. Test 1 is specified in Line 07. It uses the module-internal scan chains; these are the two scan chains specified in Line 04. Test 1 does not use a TAM. For the top-level module, this is normal; the terminals of the top-level module are SOC pins, and hence test stimuli and responses for this module do not need to be transported over a TAM first. Test 1 has 43 test patterns and dissipates 128 units of power. Test 2 of Module 0 is specified in Line 08. It (as expected) does not use a TAM, but also does not use the module-internal scan chains; perhaps this is a functional test, as opposed to a structural scan test. Test 2 has 32 test patterns and dissipates 537 units of power.

Lines 09–14 describe Module 1. Its level is 1, which means that Module 1 is embedded within Module 0. Module 1 has 10 inputs, 11 outputs, and 12 bidirectional terminals. Module 1 has four scan chains, of 20, 21, 22, and 23 scan flip flops long respectively. The X co-ordinate is given as 678 units and the Y co-ordinate is given as 123 units. Module 1 has three tests; All three use a TAM; the first two tests also use the module-internal scan chains, while Test 3 does not do this. Test 1 has 567 test patterns, and dissipates 576 units of power. Test 2 has 876 test patterns, and dissipates 275 units of power. Test 3 has 908 test patterns, and dissipates 123 units of power. It does not use the module-internal scan chains, but still depends on a module-external TAM to deliver the test data to and from the module.

Modules 2 till 6 are described in the sequel of the example. Modules 4 and 6 do not contain scan chains. Therefore, the parameter `ScanUse` becomes irrelevant for their tests. The tests of these modules also do not use a TAM, as their parameter `TamUse` is specified as '0'. Modules 4 and 6 are probably non-logic cores that are tested by means of an internal BIST.

From the order in which the modules of x847 are specified and their level information, we can establish that the design hierarchy of x847 is as depicted in Figure 1. Careful inspection of Figure 2 shows that Module 3, containing Module 4, has the same properties as Module 5, containing Module 6. This might be due to the fact that Modules 3 and 5 are equal, or, in other words, that Module 2 contains two instances of the same core (although this is not guaranteed by the benchmark format).

4 Benchmark Naming

We have adopted a naming scheme for the benchmark SOCs. Rather than simply numbering the benchmark circuits, as was done in [10], we have chosen for more meaningful names of the SOCs. The adopted naming scheme is inspired by the naming conventions of the ISCAS'85 and ISCAS'89 benchmarks [4, 6] and the naming scheme used for Philips circuits for which experimental research results have been published [11]. For the ISCAS benchmarks, the first letter is either 'c' or 's', and it indicates whether the benchmark is combinational or sequential in nature. The subsequent number lists the number of nets in the circuit and hence is for many applications representative of the size and complexity of the benchmark circuit for the problem at hand. The names of previously-published Philips circuits all started with a letter 'p', referring to Philips. The distinction between combinational and sequential circuits was superfluous, as virtually all industrial circuits are sequential anyway. The subsequent number was also the number of nets in the circuit.

The names assigned to our benchmark SOCs consist of one letter, followed by a number. The letter represents the origin of the benchmark. For example: the letter 'd' refers to Duke University, and the letter 'p' refers to Philips. Letters have been (and continue to will be) given out to benchmark contributors on a first-come-first-serve basis. This naming scheme is not guaranteed to be conflict free, but we resolve name conflicts in an ad-hoc manner whenever they occur. Three letters have a 'reserved meaning'.

- We are not using the letters 'c' and 's', in order to avoid confusion with the well-known ISCAS'85 and ISCAS'89 benchmark circuits.
- The letter 'x' is reserved for SOC benchmark contributors who want to remain anonymous.

The subsequent number is a positive integer, meant to give an indication of the test complexity of the SOC. The number is calculated based on a formula first published in [12]. Let T be 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 $l_{m(t),1}, l_{m(t),2}, \cdots, l_{m(t),s_{m(t)}}$, the binary parameters for `ScanUse` su_t and `TamUse` tu_t, and the test pattern count p_t. The formula that calculates the SOC number is as follows.

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

The scaling factor 1/10,000 is used to shorten the (often large) SOC test complexity number obtained from the formula, in order to make it easier to use.

5 Benchmark Set

The set of benchmarks are published through the Internet at the following URL: *http://www.extra.research.philips.com/itc02socbenchm/* .

Currently, the ITC'02 SOC Test Benchmark set consists of twelve benchmarks. We intend to continue to accept new contributions, as well as keep the benchmark data available.

Nine companies and institutions have contributed SOCs to the set.

- a: Analog Devices – Austin, TX, U.S.A.
- d: Duke University – Durham, NC, U.S.A.
- f: Faraday Technologies – Hsinchu, Taiwan.
- g: Jiri Gaisler and University of Stuttgart – Stuttgart, Germany.
- h: National Tsing Hua University – Hsinchu, Taiwan.
- p: Philips Electronics – Eindhoven, The Netherlands.
- q: Hewlett-Packard – Shrewsbury, MA, U.S.A.
- t: Texas Instruments – Bangalore, India.
- u: Universidade Federal do Rio Grande do Sul – Porto Alegre, RS, Brazil.

Table 1 lists the main characteristics of the SOCs. This table is organized as follows. Column 1 gives the names of the SOCs. In Column 2, the number of modules (including Module 0, the top-level SOC design itself) is listed. Column 3 lists the number of design hierarchy levels, including the top-level of Module 0; in our benchmark SOCs so far, these are either two or three. Two levels indicates there is the top-level SOC with modules embedded in it. In the case of three levels, we also have modules embedded inside embedded modules. Column 4 lists the total number of I/O terminals in the SOC; this is the sum of the I/O counts of all modules. Column 5 shows the total number of scan flip flops in the SOC; this is the sum of the scan chain lengths of all modules. Column 6 lists the total sum of the test pattern counts of all tests. Note that a high pattern count does not directly translate into a large test time. Some tests have few test patterns, but utilize very long scan chains, whereas other tests have many patterns, but do not use scan chains at all.

SOC	Number of				
	Modules	Levels	$\sum$I/Os	$\sum$SFFs	$\sum$Test Patterns
u226	10	2	376	1040	5148569
d281	9	2	2931	882	8818
d695	11	2	1845	6384	881
h953	9	2	929	4657	1100
g1023	15	2	3707	1546	2349
f2126	5	2	1597	13996	962
q12710	5	2	13167	12991	4612
p22810	29	3	4283	24723	24890
p34392	20	3	2057	20948	66349
p93791	33	3	6943	89973	22987
t512505	31	2	8663	68051	10479
a586710	8	3	3755	37656	10850894

Table 1: Some general characteristics of the ITC'02 SOC Test Benchmarks.

Table 2 lists the main characteristics of the module tests of the SOCs. Column 1 again lists the names of the SOCs. In Column 2, the summed number of tests over all modules is shown Typically, there is at least one test per module. For some SOCs, no test for Module 0 has been specified. Some SOCs have cores with more than one test per module. In Columns 3, 4, and 5, the minimum, average, and maximum number of test patterns per test is listed. Note that a high pattern count does not directly implies a large test time. The test time is dependent on the number of test patterns times the number of clock cycles it takes to load and unload one test pattern. Some tests have very many test patterns, but do not have module-internal scan chains ("`ScanChains 0:`") or do not use them ("`ScanUse 0`"), and hence their test time is still moderate. Other tests have only few test patterns, but utilize very long scan chains.

SOC	$\sum$Tests	Pattern Count		
		Minimum	Average	Maximum
u226	9	15	572063	1363968
d281	15	26	588	2048
d695	10	12	88	234
h953	8	9	138	341
g1023	14	15	168	1024
f2126	4	103	241	422
q12710	4	852	1153	1314
p22810	30	1	830	12324
p34392	21	11	3159	12336
p93791	32	11	718	6127
t512505	30	3	349	3370
a586710	7	2945	1550128	6029308

Table 2: Some test characteristics of the ITC'02 SOC Test Benchmarks.

Table 3 lists the main characteristics of the scan chains in the modules of the SOCs. Column 1 again lists the names of the SOCs. In Column 2, the summed number of scan chains over all modules is shown. In Columns 3, 4, and 5, the minimum, average, and maximum length of any of these scan chains (counted in scan flip flops) is listed.

SOC	$\sum$Scan Chains	Scan Chain Length		
		Minimum	Average	Maximum
u226	20	52	52	52
d281	34	7	26	32
d695	137	32	46	55
h953	28	21	166	348
g1023	35	9	44	84
f2126	26	318	538	1000
q12710	13	413	999	1689
p22810	196	1	126	400
p34392	63	8	332	806
p93791	522	1	172	521
t512505	64	10	1063	1669
a586710	16	2141	2354	2548

Table 3: Some scan chain characteristics of the ITC'02 SOC Test Benchmarks.

Figures 3 and 4 provide more details on the scan chains of three of the benchmark circuits. Figure 3 shows the number of scan chains per module (for those modules that do contain scan chains) for SOCs p22810, p34392, and p93791. For the same SOCs, Figure 4 shows the minimum (light blue), average (dark red), and maximum (light yellow) scan chains lengths per module as stacked bar charts. From these figures, we can judge that the benchmark set contains quite diverse SOCs. SOC p22409 contains 29 modules, of which 22 (= 76%) contain scan chains, whereas for SOC p34392 only 4 out of the 20 modules (= 20%) contain scan chains. The number of scan chains per module varies widely for SOC p22409, while almost all modules of SOC p93791 have 46 scan chains. For some SOCs, the scan chains per module are nicely balanced, such that there is little variation in minimum, average, and maximum scan chain length, whereas for other SOCs that is not the case. These characteristics can have a large impact on the effectiveness of test architecture optimization techniques [13].

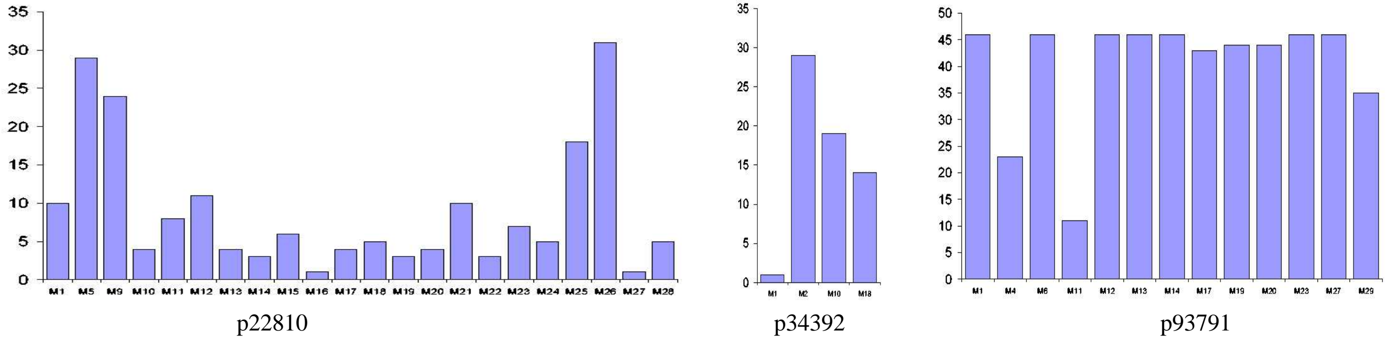

Figure 3: Number of scan chains per module for SOCs p22810, p34392, and p93791.

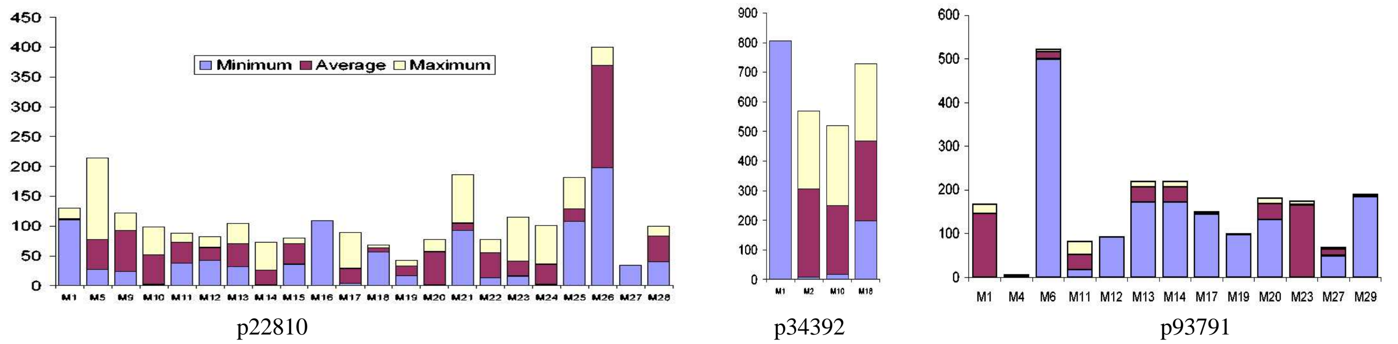

Figure 4: Minimum, average, and maximum scan chain lengths per module for SOCs p22810, p34392, and p93791.

6 Classification of Problems and Prior Work in SOC Test Automation

The SOC benchmarks presented in this paper are intended to be used for research that addresses a wide range of problems in modular testing of SOCs. In this section, we first present a classification of several of such problems in SOC test automation. We then present a corresponding listing of related prior work. Several of the papers in the prior work listing already include experimental results for circuits similar to the SOC benchmarks presented in this paper. However, very few papers actually compare their solutions with competing research work using a common set of benchmarks. The proposed benchmarks are intended to fill this vital need; they will serve as experimental vehicles that can be used to compare and contrast SOC test automation algorithms and tools.

Figure 5 presents a graphical classification of several problems in SOC test automation. Note that this classification is by no means inclusive of all the problems in SOC test automation; our list is not even inclusive of all the problems for which the proposed benchmarks can be used. The problems presented in this classification have been collected by surveying the literature and are meant to be a stepping-stone to identify areas of future research. The problems are classified into three broad categories: (i) core test wrappers, (ii) test access mechanisms (TAMs), and (iii) test scheduling. The category 'TAM Design + Scheduling' is derived by combining problems from the latter two categories. The 'wrappers' category is divided into wrapper design and wrapper optimization subcategories. Here, we use the term *design* to denote actual proposals for wrappers, such as the 'test collar' [14] or the IEEE P1500 SECT standard-under-development [15]. The term *optimization* is used to refer to the process of determining the parameters of the wrapper for a specific core, such that certain objectives are met while fulfilling certain constraints, e.g., test time minimization under a maximum TAM width constraint. An example of a wrapper optimization approach can be found in [16]. The 'TAMs' category is similarly divided into design and optimization subcategories. For the first three broad categories of problems presented in Figure 5, we list optimization objectives and constraints. For example, an objective of TAM optimization is to minimize testing time, while a constraint of test scheduling is to ensure that the maximum power consumption limit is not exceeded.

We now list brief descriptions of related prior work in SOC test automation. The numbers in this list correspond to the circled numbers in Figure 5.

1. **Test wrapper design**. The test wrapper and TAM model of SOC test access architectures was presented in [17]. A 'test collar' was proposed in [14] to be used as a test wrapper for cores and to complement the test bus model for TAMs. The TESTSHELL proposed in [18] is a similar wrapper, meant to be used in conjunction with the TESTRAIL. The IEEE P1500 SECT Working Group is working towards an industry-wide standard, but scalable wrapper [19].

2. **Test wrapper optimization**. The issue of efficient deserialization of test data by means of using balanced wrapper scan chains was discussed in [20]. Heuristics for designing balanced wrapper scan chains, based on approximation algorithms for the well-known Bin Design problem were presented in [16]. The *Design_wrapper* algorithm proposed in [21] has two priori-

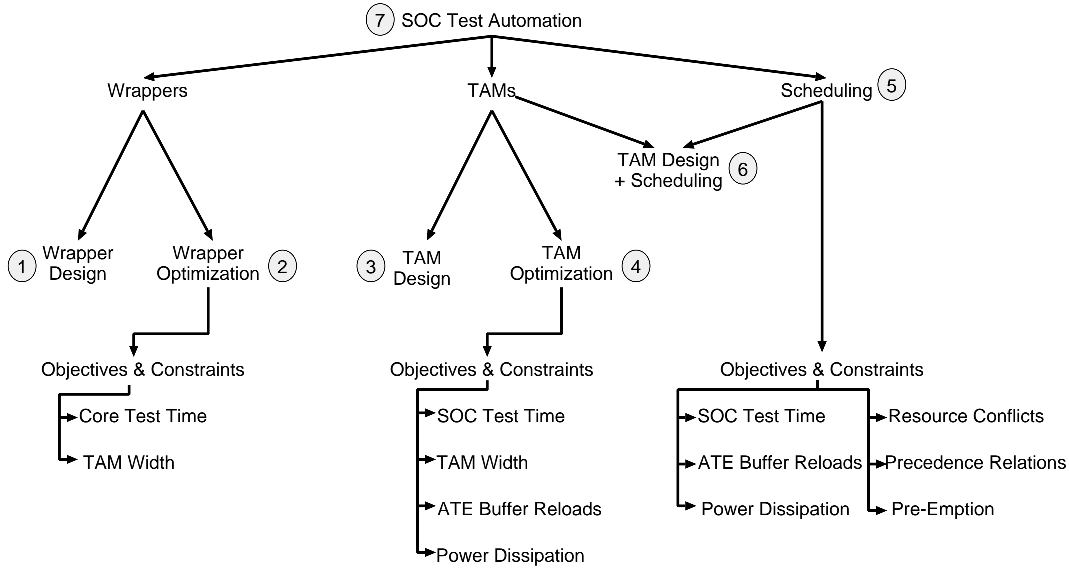

Figure 5: A classification of problems in SOC test automation.

ties: (i) minimizing core testing time , and (ii) minimizing the TAM width required for the test wrapper. This algorithm is based on the Best Fit Decreasing (BFD) heuristic for the Bin Packing problem.

3. **TAM design**. A number of TAM designs have been proposed in the literature. These include multiplexed access [22], partial isolation rings [23], core transparency [24, 25, 26, 27], and a hierarchical TAM structure [20]. Bus-based TAM proposals include reuse of the existing system bus [28], a dedicated scalable tri-statable test bus [14], a scalable daisychain-able architecture called TESTRAIL [18], and a P1500-compatible TAM known as CAS-BUS [29]. In addition, the use of a single TAM design to test both cores with TAPs and cores with test wrappers was proposed in [30]. Dedicated and scalable TAMs appear to be the most promising. In [31, 32], a hierarchical framework for BIST scheduling, data pattern delivery and diagnosis was proposed.
4. **TAM optimization**. Several novel TAM architectures (i.e., multiplexing, daisy chaining and distribution) were proposed in [33]. The relationship between testing time and TAM widths using ILP was examined in [34, 35], and TAM width optimization under power and routing constraints was studied in [36, 37]. A genetic algorithm was used for TAM optimization in [38].
5. **Test scheduling**. Several techniques for SOC test scheduling have been proposed in the literature. These include combinatorial optimization [39], test reordering for a large batch of ICs [40], test protocol scheduling [41, 42], integer linear programming [43], and power-constrained scheduling [44, 45, 46, 47, 48, 49]. Methods to incorporate precedence and power constraints in a preemptive test schedule were presented in [50].
6. **Integrated TAM optimization and test scheduling**. Integrated TAM optimization and test scheduling was first attempted in [51, 52]. In [53, 54], rectangle packing approaches to schedule tests were reported. Related recent work can be found in [55].
7. **Integrated Wrapper/TAM co-optimization and test scheduling**. The first integrated method for wrapper/TAM co-optimization was proposed in [21]. Exact methods and enumeration were used to optimize a Test Bus Architecture, assuming an arbitrary, but sequentially-ordered schedule of the cores per test bus. In [56], efficient heuristic algorithms to solve the same problem were presented. A rectangle packing approach was reported in [57], and extended with precedence, power dissipation, and preemption constraints in [58]. Effective and efficient optimization algorithms for the TestRail Architecture were presented in [13].

Additionally, a more complete listing of current and prior work can be found at [3].

Several open problems in SOC test automation are listed here as follows.

- Minimization of the number of ATE buffer reloads during the test for an SOC.
- Scheduling of interconnect tests (ExTest).
- Hierarchical TAM optimization – multi-level TAMs.
- Test scheduling for hierarchical SOCs.
- TAM width optimization to minimize routing overhead.

The proposed benchmarks are especially suitable for use in research that addresses these problems.

7 Conclusion

In this paper, we have presented the *ITC'02 SOC Test Benchmarks*. This set of benchmarks is intended to stimulate research into new methods and tools for modular testing of SOCs and to enable objective comparison of the research results. The paper describes the benchmark format and naming scheme, and illustrates this by means of

an example. The paper describes the current set of benchmarks and some of their characteristics. The paper also provides a classification of addressed and open research problems for which the benchmarks can serve as test cases. The benchmark set currently contains twelve SOCs, but might still grow in the future. The benchmark SOCs are freely available to anyone and can be downloaded from *http://www.extra.research.philips.com/itc02socbenchm/*.

Acknowledgements

We thank the following persons and organizations for their contribution to the SOC benchmark set: Luis Basto, Analog Devices (Austin, TX, U.S.A); Chuang Cheng, Faraday Technologies (Hsinchu, Taiwan); Erika Cota, Universidade Federal do Rio Grande do Sul (Porto Alegre, Brazil); Rainer Dorsch, University of Stuttgart (Stuttgart, Germany); Graeme Francis, Philips Semiconductors (Southampton, United Kingdom); Chul Young Lee, Hewlett-Packard (Shrewsbury, MA, U.S.A.); Rubin Parekhji, Texas Instruments (Bangalore, India); Erwin Waterlander, Philips Research (Eindhoven, The Netherlands); Cheng-Wen Wu, National Tsing Hua University (Hsinchu, Taiwan). We thank Sandeep Kumar Goel of Philips Research for building a parser for the .soc benchmark format. Finally, we thank both Sandeep Kumar Goel and Harald Vranken for their useful comments on a draft version of this paper.

References

[1] Erik Jan Marinissen and Yervant Zorian. Challenges in Testing Core-Based System ICs. *IEEE Communications Magazine*, 37(6):104–109, June 1999.

[2] Yervant Zorian, Erik Jan Marinissen, and Sujit Dey. Testing Embedded-Core Based System Chips. In *Proceedings IEEE International Test Conference (ITC)*, pages 130–143, Washington, DC, October 1998.

[3] Erik Jan Marinissen. The TECS Bibliography Homepage. http://www.extra.research.philips.com/itc02socbenchm/bib/.

[4] Franz Brglez and Hideo Fujiwara. A Neutral Netlist of 10 Combinational Benchmark Circuits and a Target Simulator in FORTRAN. In *Proceedings International Symposium on Circuits and Systems (ISCAS)*, pages 695–698, Kyoto, Japan, May 1985.

[5] ISCAS'85 Benchmarks Web Site. http://www.cbl.ncsu.edu/www/CBL_Docs/iscas85.html.

[6] F. Brglez, D. Bryan, and K. Kozminski. Combinational Profi les of Sequential Benchmark Circuits. In *Proceedings International Symposium on Circuits and Systems (ISCAS)*, pages 1924–1934, Portland, OR, May 1989.

[7] ISCAS'89 Benchmarks Web Site. http://www.cbl.ncsu.edu/www/CBL_Docs/iscas89.html.

[8] Scott Davidson. Characteristics of the ITC'99 Benchmark Circuits. In *IEEE International Test Synthesis Workshop (ITSW)*, Santa Barbara, CA, March 1999.

[9] Scott Davidson. ITC'99 Benchmark Circuits – Preliminary Results. In *Proceedings IEEE International Test Conference (ITC)*, page 1125, Atlantic City, NJ, September 1999.

[10] ITC'99 Benchmarks Web Site. http://www.cerc.utexas.edu/itc99-benchmarks/bench.html.

[11] Gundolf Kiefer, Harald Vranken, Erik Jan Marinissen, and Hans-Joachim Wunderlich. Application of Deterministic Logic BIST on Industrial Circuits. In *Proceedings IEEE International Test Conference (ITC)*, pages 105–114, Atlantic City, NJ, October 2000.

[12] Vikram Iyengar, Krishnendu Chakrabarty, and Erik Jan Marinissen. Test Wrapper and Test Access Mechanism Co-Optimization for System-on-Chip. In *Proceedings IEEE International Test Conference (ITC)*, pages 1023–1032, Baltimore, MD, October 2001.

[13] Sandeep Kumar Goel and Erik Jan Marinissen. Cluster-Based Test Architecture Design for System-on-Chip. In *Proceedings IEEE VLSI Test Symposium (VTS)*, pages 259–264, Monterey, CA, April 2002.

[14] Prab Varma and Sandeep Bhatia. A Structured Test Re-Use Methodology for Core-Based System Chips. In *Proceedings IEEE International Test Conference (ITC)*, pages 294–302, Washington, DC, October 1998.

[15] Erik Jan Marinissen, Rohit Kapur, and Yervant Zorian. On Using IEEE P1500 SECT for Test Plug-n-Play. In *Proceedings IEEE International Test Conference (ITC)*, pages 770–777, Atlantic City, NJ, October 2000.

[16] Erik Jan Marinissen, Sandeep Kumar Goel, and Maurice Lousberg. Wrapper Design for Embedded Core Test. In *Proceedings IEEE International Test Conference (ITC)*, pages 911–920, Atlantic City, NJ, October 2000.

[17] Yervant Zorian, Erik Jan Marinissen, and Sujit Dey. Testing Embedded-Core-Based System Chips. *IEEE Computer*, 32(6):52–60, June 1999.

[18] Erik Jan Marinissen et al. A Structured And Scalable Mechanism for Test Access to Embedded Reusable Cores. In *Proceedings IEEE International Test Conference (ITC)*, pages 284–293, Washington, DC, October 1998.

[19] Erik Jan Marinissen, Yervant Zorian, Rohit Kapur, Tony Taylor, and Lee Whetsel. Towards a Standard for Embedded Core Test: An Example. In *Proceedings IEEE International Test Conference (ITC)*, pages 616–627, Atlantic City, NJ, September 1999.

[20] Tapan J. Chakraborty, Sudipta Bhawmik, and Chen-Huan Chiang. Test Access Methodology for System-On-Chip Testing. In *Digest of Papers of IEEE International Workshop on Testing Embedded Core-Based Systems (TECS)*, pages 1.1–1–7, Montreal, Canada, May 2000.

[21] Vikram Iyengar, Krishnendu Chakrabarty, and Erik Jan Marinissen. Co-Optimization of Test Wrapper and Test Access Architecture for Embedded Cores. *Journal of Electronic Testing: Theory and Applications*, 18(2):213–230, April 2002.

[22] Venkata Immaneni and Srinivas Raman. Direct Access Test Scheme - Design of Block and Core Cells for Embedded ASICs. In *Proceedings IEEE International Test Conference (ITC)*, pages 488–492, September 1990.

[23] Nur Touba and Bahram Pouya. Using Partial Isolation Rings to Test Core-Based Designs. *IEEE Design & Test of Computers*, 14(4):52–59, December 1997.

[24] Erik Jan Marinissen, Krijn Kuiper, and Clemens Wouters. Test Protocol Expansion in Hierarchical Macro Testing. In *Proceedings IEEE European Test Conference (ETC)*, pages 28–36, Rotterdam, The Netherlands, April 1993.

[25] Indradeep Ghosh, Sujit Dey, and Niraj K. Jha. A Fast and Low Cost Testing Technique for Core-based System-on-Chip. In *Proceedings ACM/IEEE Design Automation Conference (DAC)*, pages 542–547, San Francisco, CA, June 1998. Association for Computing Machinery, Inc.

[26] Krishnendu Chakrabarty, Rajatish Mukherjee, and Andrew S. Exnicios. Synthesis of Transparent Circuits for Hierarchical and System-on-a-Chip Test. In *Proceedings IEEE International Conference on VLSI Design (ICVD)*, pages 431–436, Bangalore, India, January 2001.

[27] Tomokazu Yoneda and Hideo Fujiwara. A DfT Method for Core-Based Systems-on-a-Chip based on Consective Testability. In *Proceedings IEEE Asian Test Symposium (ATS)*, pages 193–198, Kyoto, Japan, November 2001.

[28] Peter Harrod. Testing Reusable IP - A Case Study. In *Proceedings IEEE International Test Conference (ITC)*, pages 493–498, Atlantic City, NJ, September 1999.

[29] Mounir Benabdenbi, Walid Maroufi , and Meryem Marzouki. CAS-BUS: A Scalable and Reconfi gurable Test Access Mechanism for Systems on a Chip. In *Proceedings Design, Automation, and Test in Europe (DATE)*, pages 141–145, Paris, France, March 2000.

[30] Mounir Benabdenbi, Walid Maroufi, and Meryem Marzouki. Testing TAPed Cores and Wrapped Cores With The Same Test Access Mechanism. In *Proceedings Design, Automation, and Test in Europe (DATE)*, pages 150–155, Munich, Germany, March 2001.

[31] Alfredo Benso, Silvia Cataldo, Silvia Chiusano, Paolo Prinetto, and Yervant Zorian. HD-BIST: A Hierarchical Framework for BIST Scheduling and Diagnosis in SOCs. In *Proceedings IEEE International Test Conference (ITC)*, pages 1038–1044, Atlantic City, NJ, September 1999.

[32] Alfredo Benso et al. HD^2BIST: A Hierarchical Framework for BIST Scheduling, Data Patterns Delivering and Diagnosis in SoCs. In *Proceedings IEEE International Test Conference (ITC)*, pages 892–901, Atlantic City, NJ, October 2000.

[33] Joep Aerts and Erik Jan Marinissen. Scan Chain Design for Test Time Reduction in Core-Based ICs. In *Proceedings IEEE International Test Conference (ITC)*, pages 448–457, Washington, DC, October 1998.

[34] Krishnendu Chakrabarty. Design of System-on-a-Chip Test Access Architectures Using Integer Linear Programming. In *Proceedings IEEE VLSI Test Symposium (VTS)*, pages 127–134, Montreal, Canada, April 2000.

[35] Krishnendu Chakrabarty. Optimal Test Access Architectures for System-on-a-Chip. *ACM Transactions on Design Automation of Electronic Systems*, 6(1):26–49, January 2001.

[36] Krishnendu Chakrabarty. Design of System-on-a-Chip Test Access Architectures Under Place-and-Route and Power Constraints. In *Proceedings ACM/IEEE Design Automation Conference (DAC)*, pages 432–437, Los Angeles, CA, June 2000.

[37] Vikram Iyengar and Krishnendu Chakrabarty. Test Bus Sizing for System-on-a-Chip. *IEEE Transactions on Computers*, 51:449–459, May 2002.

[38] Zahra sadat Ebadi and Andre Ivanov. Design of an Optimal Test Access Architecture Using a Genetic Algorithm. In *Proceedings IEEE Asian Test Symposium (ATS)*, pages 205–210, Kyoto, Japan, November 2001.

[39] Makoto Sugihara, Hiroshi Date, and Hiroto Yasuura. A Novel Test Methodology for Core-Based System LSIs and a Testing Time Minimization Problem. In *Proceedings IEEE International Test Conference (ITC)*, pages 465–472, Washington, DC, October 1998.

[40] Wanli Jiang and Bapiraju Vinnakota. Defect-Oriented Test Scheduling. In *Proceedings IEEE VLSI Test Symposium (VTS)*, pages 433–438, Dana Point, CA, April 1999.

[41] Erik Jan Marinissen and Maurice Lousberg. The Role of Test Protocols in Testing Embedded-Core-Based System ICs. In *Proceedings IEEE European Test Workshop (ETW)*, pages 70–75, Konstanz, Germany, May 1999.

[42] Erik Jan Marinissen. The Role of Test Protocols in Automated Test Generation for Embedded-Core-Based System ICs. *Journal of Electronic Testing: Theory and Applications*, 18(4), August 2002.

[43] Krishnendu Chakrabarty. Test Scheduling for Core-Based Systems Using Mixed-Integer Linear Programming. *IEEE Transactions on Computer-Aided Design*, 19(10):1163–1174, October 2000.

[44] Yervant Zorian. A Distributed BIST Control Scheme for Complex VLSI Devices. In *Proceedings IEEE VLSI Test Symposium (VTS)*, pages 6–11, Princeton, NJ, April 1993.

[45] R.M. Chou, K.K. Saluja, and V.D. Agrawal. Scheduling Tests for VLSI Systems under Power Constraints. *IEEE Transactions on VLSI Systems*, Vol. 5(No. 2):175–185, June 1997.

[46] Valentin Mureşan et al. A Comparison of Classical Scheduling Approaches in Power-Constrained Block-Test Scheduling. In *Proceedings IEEE International Test Conference (ITC)*, pages 882–891, Atlantic City, NJ, October 2000.

[47] Paul Rosinger, Bashir Al-Hashimi, and Nicola Nicolici. Power Constrained Test Scheduling Using Power Profile Manipulation. In *Proceedings International Symposium on Circuits and Systems (ISCAS)*, volume V, pages V251–V254, May 2001.

[48] Erik Larsson and Zebo Peng. Test Scheduling and Scan-Chain Division Under Power Constraint. In *Proceedings IEEE Asian Test Symposium (ATS)*, pages 259–264, Kyoto, Japan, November 2001.

[49] Marie-Lise Flottes, Julien Pouget, and Bruno Rouzeyre. Sessionless Test Scheme: Power-Constrained Test Scheduling for System-on-a-Chip. In *Proceedings IFIP International Conference on Very Large Scale Integration (VLSI-SOC)*, pages 105–110, Montpellier, France, December 2001.

[50] Vikram Iyengar and Krishnendu Chakrabarty. Precedence-Based, Preemptive, and Power-Constrained Test Scheduling for System-on-a-Chip. In *Proceedings IEEE VLSI Test Symposium (VTS)*, pages 368–374, Marina del Rey, CA, May 2001.

[51] Erik Larsson and Zebo Peng. An Integrated System-on-Chip Test Framework. In *Proceedings Design, Automation, and Test in Europe (DATE)*, pages 138–144, Munich, Germany, March 2001.

[52] Mehrdad Nourani and Chris Papachristou. An ILP Formulation to Optimize Test Access Mechanism in System-on-Chip Testing. In *Proceedings IEEE International Test Conference (ITC)*, pages 902–910, Atlantic City, NJ, October 2000.

[53] Yu Huang et al. Resource Allocation and Test Scheduling for Concurrent Test of Core-Based SOC Design. In *Proceedings IEEE Asian Test Symposium (ATS)*, pages 265–270, Kyoto, Japan, November 2001.

[54] Yu Huang et al. Constraint-Driven Pin Mapping for Concurrent SOC Testing. In *Proceedings IEEE Asia South Pacific Design Automation Conference (ASP-DAC)*, Bangelore, India, January 2002.

[55] Sandeep Koranne. On Test Scheduling for Core-Based SOCs. In *Proceedings IEEE Asia South Pacific Design Automation Conference (ASP-DAC)*, pages 505–510, Bangelore, India, January 2002.

[56] Vikram Iyengar, Krishnendu Chakrabarty, and Erik Jan Marinissen. Efficient Wrapper/TAM Co-Optimization for Large SOCs. In *Proceedings Design, Automation, and Test in Europe (DATE)*, pages 491–498, Paris, France, March 2002.

[57] Vikram Iyengar, Krishnendu Chakrabarty, and Erik Jan Marinissen. On Using Rectangle Packing for SOC Wrapper/TAM Co-Optimization. In *Proceedings IEEE VLSI Test Symposium (VTS)*, pages 253–258, Monterey, CA, April 2002.

[58] Vikram Iyengar, Krishnendu Chakrabarty, and Erik Jan Marinissen. Integrated Wrapper/TAM Co-Optimization, Constraint-Driven Test Scheduling, and Tester Data Volume Reduction for SOCs. In *Proceedings ACM/IEEE Design Automation Conference (DAC)*, pages 685–690, New Orleans, LO, June 2002.