Efficient Test Solutions for Core-Based Designs

Erik Larsson, *Member, IEEE*, Klas Arvidsson, Hideo Fujiwara, *Fellow, IEEE*, and Zebo Peng, *Senior Member, IEEE*

***Abstract*—A test solution for a complex system requires the design of a test access mechanism (TAM), which is used for the test data transportation, and a test schedule of the test data transportation on the designed TAM. An extensive TAM will lead to lower test-application time at the expense of higher routing costs, compared to a simple TAM with low routing cost but long testing time. It is also possible to reduce the testing time of a testable unit by loading the test vectors in parallel, thus increasing the parallelization of a test. However, such a test-time reduction often leads to higher power consumption, which must be kept under control since exceeding the power budget could damage the system under test. Furthermore, the execution of a test requires resources and concurrent execution of tests may not be possible due to resource or other conflicts. In this paper, we propose an integrated technique for test scheduling, test parallelization, and TAM design, where the test application time and the TAM routing are minimized, while considering test conflicts and power constraints. The main features of our technique are the efficiency in terms of computation time and the flexibility to model the system's test behavior, as well as the support for the testing of interconnections, unwrapped cores and user-defined logic. We have implemented our approach and made several experiments on benchmarks as well as industrial designs in order to demonstrate that our approach produces high-quality solution at low computational cost.**

***Index Terms*—Scan-chain partitioning, system-on-chip (SOC) testing, test access mechanism design, test data transportation, test scheduling, test solutions.**

I. Introduction

THE ADVANCE in design methodologies and semiconductor process technologies has led to the development of systems with excessive functionality implemented on a single die, called system-on-chip (SOC). In a core-based design approach, a set of cores, i.e., predefined and preverified design modules, is integrated into a system using user-defined logic (UDL) and interconnections. In this way, complex systems can be efficiently developed. However, the complexity in the systems leads to high test data volumes and the development of a test solution must therefore consider the following interdependent problems:

- how to design an infrastructure for the transportation of test data in the system, a test access mechanism (TAM);
- how to design a test schedule to minimize test time, considering test conflicts and power constraints.

The testable units in an SOC design are the cores, the UDL, and the interconnections. The cores are usually delivered with predefined test methods and test sets, while the test sets for UDL and interconnections are to be generated prior to test scheduling and TAM design. The test vectors, forming the test sets for each testable unit, are stored or created in some test source, and their test responses are stored or analyzed in some test sink. The TAM is the connection between the test sources, the testable units and the test sinks. The test-application time can be minimized by applying several test sets concurrently; however, test conflicts, limitations, and test-power consumption must be considered.

The workflow when developing an SOC test solution can mainly be divided into two consecutive parts: an early design space exploration followed by an extensive optimization for the final solution. For the former, we have proposed a technique for integrated test scheduling and TAM design to minimize test time and TAM cost [29], [32]. The advantage of the technique is its low computational cost making it useful for iteratively use in the early design space exploration phase. For extensive optimization of the final solution, we have proposed a technique based on simulated annealing, which is used only a few times, justifying its high computational cost [30], [32]. We have also proposed an integrated test scheduling and scan-chain-partitioning (test parallelization) technique under power constraints [31]. The test-parallelization problem is, for a testable unit with variable test time such as scan-tested cores, to determine the number of scan chains to be loaded concurrently, i.e., to determine the test time for each testable unit in such a way that the system's total test time is minimized while considering test-power limitations.

In this paper, we propose a technique to integrate test scheduling, test parallelization (scan-chain partitioning) and TAM design with the objective to minimize the test application time and the TAM routing cost while considering test conflicts and power constraints. The aim with our approach is to reduce the gap between the design space exploration and the extensive optimization, i.e., to produce a high quality solution in respect to test time and TAM cost at a relatively low computational cost.

The features of our proposed approach are that we support:

- the testing of interconnections;
- the testing of UDL;
- the testing of unwrapped cores;
- the consideration of memory limitations at test sources;
- the consideration of bandwidth limitations on test sources and test sinks;
- embedding cores in core.

Manuscript received August 19, 2002; revised February 7, 2003 and June 18, 2003. This work was supported in part by the Japan Society for the Promotion of Science (JSPS) under Grant P01735 and in part by the Swedish National Program on Socware. A shorter version of the paper was presented at the Asian Test Symposium, Tamuning, Guam, November 18–20, 2002. This paper was recommended by Associate Editor K. Chakrabarty.

E. Larsson is with the Embedded Systems Laboratory, Linköpings Universitet, SE-581 83 Linköpings, Sweden, and also with the Computer Design and Test Laboratory, Nara Institute of Science and Technology, Nara 630-0101, Japan (e-mail: erila@ida.liu.se; zebpe@ida.liu.se).

K. Arvidsson and Z. Peng are with the Embedded Systems Laboratory, Linköpings Universitet, SE-581 83 Linköpings, Sweden (e-mail: zebpe@ida.liu.se).

H. Fujiwara is with the Computer Design and Test Laboratory, Nara Institute of Science and Technology, Nara 630-0101, Japan (e-mail: fujiwara@is.aist-nara.ac.jp).

Digital Object Identifier 10.1109/TCAD.2004.826560

0278-0070/04$20.00 © 2004 IEEE

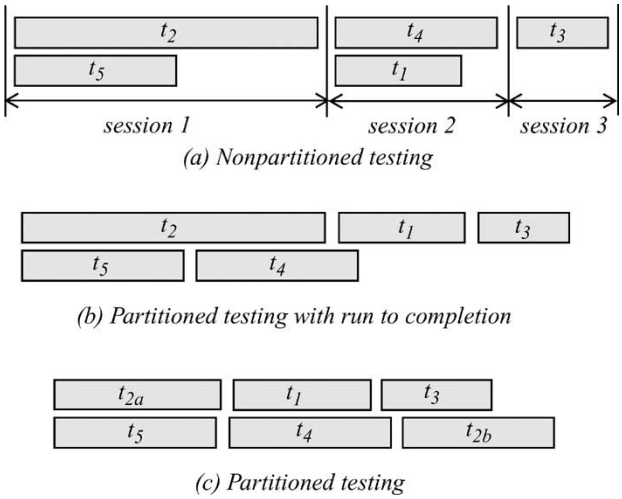


Fig. 1. Scheduling approaches.

We have implemented our technique and performed experiments on several benchmarks including a large industrial design called Ericsson, which is tested by 170 test sets. The experimental results demonstrate that we can deal with systems tested with different test methods; our approach is not limited to scan-based systems.

The organization of the paper is as follows. An introduction to the background and an overview of related work are given in Section II. The considered test problems are discussed and described in Section III. The system model is defined in Section IV, and our integrated test scheduling, test parallelization and TAM design technique is presented in Section V. The paper is concluded with experimental results in Section VI and conclusions in Section VII.

II. Background and Related Work

The test-application time when testing a system can be minimized by scheduling the execution of the test sets as concurrently as possible. The basic idea in test scheduling is to determine when each test set should be executed, and the main objective is to minimize the test application time. However, various conflicts and limitations must be considered. For instance, only one test set can be applied at any time to each testable unit. Power constraints must also be carefully considered otherwise the system under test can be damaged. The scheduling techniques can be classified using a scheme by Craig *et al.* [7] into:

- nonpartitioned testing,
- partitioned testing with run to completion, and
- partitioned testing.

The differences among the techniques are illustrated with five test sets $(t_1, \ldots, t_5)$ in Fig. 1, where the length of the rectangles corresponds to the test time of respective test sets. In nonpartitioned testing, Fig. 1(a), test sets are grouped into sessions and new tests are allowed to start only when all test sets in the preceding session are completely executed. A test-scheduling approach based on partitioned testing with run to completion does not group tests into sessions, and new tests are therefore allowed to start at any time [Fig. 1(b)]. And, finally, in partitioned testing or preemptive testing, a test can be interrupted and resumed at a later point, as test t_2 in Fig. 1(c), which is split into two partitions.

A set of test vectors is called a test set, and a system is usually tested by applying a number of test sets. For every test set, one test source and one test sink are required. The test source is where the test sets are stored or produced. A test source can be placed either on- or off-chip. The test sink is where the test response, produced by the testable unit when a test vector is applied, is stored or analyzed. Test sinks can, as test sources, be placed either on-chip or off-chip. If both the test source and the test sink for a particular testable unit are placed on-chip, it is common to refer to it as built-in self-test (BIST). An example of an on-chip test source is a linear-feedback shift-register (LFSR) or a memory. An example of an off-chip test source is an automatic test equipment (ATE). The main advantage of using an ATE as a test source and test sink is that a relatively small test set can be used for each testable unit. However, among the disadvantages are the slow speed of an ATE and its limited memory capacity [10]. An on-chip test source such as an LFSR, on the other hand, does not require an extensive global test infrastructure, which is especially true if each testable unit has its dedicated LFSR. The disadvantage with an LFSR is that usually a relatively large test set is required, which leads to long testing times and also more activity (power consumption) in the system.

The test sources and the sinks can be shared among several testable units. And every testable unit is tested by one or more test sets. A system may contain several test sources and test sinks. A test infrastructure (TAM) is used to connect the test sources, the testable units and the test sinks. The TAM is used for the transportation of test vectors from a test source to a testable unit and test responses from a testable unit to a test sink.

Zorian has proposed a test scheduling technique based on nonpartition testing [see Fig. 1(a)], for systems where each testable unit has its dedicated on-chip test source and on-chip test sink [44]. In the approach, a test set is assigned a fixed test time and a fixed test-power consumption value. The objective is to minimize the total test application time and the routing of control lines while making sure that the total test-power consumption at any time is below a given limit. The minimization of control lines is achieved by grouping tests based on the floor-plan in such a way that testing of neighboring cores are scheduled in the same test session. The advantage with the grouping is that the control lines can be shared among all test sets executed in the same session. Recently, Wang *et al.* proposed a test scheduling technique based on partitioned testing with run to completion for memories with dedicated BIST [43].

An analytic test-scheduling approach, also for nonpartitioned testing, was proposed by Chou *et al.* [5], where, as in Zorian's approach, each test set is assigned a fixed test-time and a fixed test-power value. Test conflicts are modeled in a general way using a resource graph. Based on the resource graph, a test compatibility graph is generated and a covering table is used to determine the tests scheduled in the same test session. Muresan *et al.* [39] have proposed a test scheduling technique with the same assumptions as Chou *et al.* and to allow a higher degree of

flexibility in the scheduling process partitioned testing with run to completion is used.

In all the above approaches, each testable unit has one dedicated test set with a fixed test time. Sugihara *et al.* proposed a technique for the selection of test sets for each testable unit where each testable unit can be tested by one test set using an off-chip test source and an off-chip test sink as well as one test set using a dedicated on-chip test source and a dedicated on-chip test sink [41]. The objective is to find a tradeoff between the number of test vectors in on-chip resources (test source and test sink) and off-chip resources (test source and test sink). The sharing of test resources may introduce conflicts if a test resource can only generate test patterns for a testable unit at a time. Chakrabarty also proposed a test scheduling technique for systems tested by two test sets, one BIST test set and one stored at the ATE [2], [3]. Chakrabarty also considered the conflicts that appears when sharing the test bus for test data transportation. Furthermore, the sharing of BIST resources among testable units is considered.

A test infrastructure is used for the transportation of test data, that is test vectors and test responses. The advanced microcontroller bus architecture (AMBA) is an approach where all test sets are scheduled in a sequence on a single bus [9]. Another bus approach is proposed by Varma and Bhatia [42]. Instead of having all TAM wires in a single bus, Varma and Bhatia propose a technique where several set of wires form several test buses. The tests on each test bus are, as in the case with AMBA, scheduled in a sequence. However, tests on different buses are executed concurrently. Aerts and Marinissen have also proposed three architectures, *multiplexing* where all tests are scheduled in a sequence, *distributed* where all tests are scheduled concurrently on their dedicated TAM wires, and *daisy-chain* where all tests are scheduled concurrently and as soon as a core is finished with its testing, the core is by-passed using a clocked buffer [1]. A common drawback with scheduling tests in a sequence is that the testing of interconnection is a bit cumbersome.

The advantage with scheduling the tests in a sequence is that only one test set is active at a time and there is only switching in one testable unit at a time, which reduces the test-power consumption. In an concurrent approach, the testing of several cores can be performed at the same time. In the distributed architecture, the testing of all cores are started at the same time, which means that all cores are activated simultaneously, resulting in a high test-power consumption. In the daisy-chain approach, all cores are also scheduled to start at the same time and the test vectors are pipelined through the cores. The idea of daisy-chain tests is efficient from a test-time perspective, however, from a test-power consumption perspective it results in a high switching activity. Saxena *et al.* have proposed a technique to reduce the test-power consumption in scan-based designs by using a gated subchain scheme [40]. Experimental results indicate that comparing an original design and a design with gated subchains, the test time remains the same but the test-power consumption is reduced by the number of gated sub chains.

A core test wrapper is the interface between a core and the TAM. A standard core wrapper such as the proposed P1500 or the TestShell can be in four modes, normal operation, internal core test, external test, and bypass [14], [36]. The tests in a system can be grouped into wrapped core tests and unwrapped core tests. A wrapped core test is a test at a core equipped with a dedicated interface (a wrapper) to the TAM and an unwrapped core test is a test at a core that does not have a dedicated wrapper. A new conflict appears, namely test wrapper conflict. Several approaches have been proposed for wrapped core tests. Iyengar *et al.* proposed a technique for core tests where a fixed TAM bandwidth is assumed to have been partitioned into a set of fixed TAMs and the problem is to assign cores to the TAMs in such a way that the total test-application time is minimized [15]. The tests on each TAM are scheduled in a sequence and the tests can be assigned to any of the TAM's in the system. In order to make the approach applicable to large industrial designs, Iyengar *et al.* have proposed the use of an heuristic instead of integer linear programming (ILP) [16].

Several approaches by Iyengar *et al.* [17], Goel and Marinissen [20], Goel and Marinissen [21], Goel and Marinissen [22], Huang *et al.* [12], Koranne [25], and Koranne and Iyengar [26] have been proposed for the assignment of TAM wires to each core test. Hsu *et al.* [11] and Huang *et al.* [13] also proposed techniques for TAM wire assignment under power constraints for tests with fixed power consumption and variable test times. Iyengar *et al.* proposed a technique where hierarchical conflicts are considered [19]. An approach for TAM design and test scheduling is proposed by Cota *et al.* [6]. The test data can be transported on dedicated TAM as well as on the functional bus. To further explore the design space, the approach allows redesign and extensions of the functional bus.

III. Test Problems

In this section, we describe the test problems we are considering and their modeling.

A. Test Time

In this paper, we use a test-time model that assumes within a given range a linear dependency between the test time and the number of TAM wires assigned to a testable unit. In our model, we assume that the designer specifies a bandwidth range for each core. It means that for each testable unit a bandwidth is to be selected, which is within the minimal bandwidth and the maximal bandwidth range and the test time contra TAM bandwidth within the given range is linear.

The test time for executing a test at a testable unit is defined by the time required for applying the test vectors. For some test methods, the test time is fixed, while for other test methods, such as scan-based testing, the testing time can be modified. A modification is achieved by test parallelization. For instance assume a core with a number of scan chains which form a single chain connected to a single TAM wire. The testing of the core is performed by shifting in a test vector and when the scan chains are filled, a capture cycle is applied, and then the test response is shifted out. A major part of the testing time is therefore consumed by the shift process. In order to reduce the shifting time, a new test vector can be shifted in at the same time as the test response from the previous test vector is shifted out. To further reduce the time consumed due to shifting, the scan chains can be connected into several wrapper chains where each wrapper

TABLE I
CHARACTERISTICS FOR THE SCAN-BASED CORES IN DESIGN P93791

Core	Test vectors	Inputs	Outputs	Bidirs	Scan chains	Shortest scan-chain	Longest scan-chain
1	409	109	32	72	46	1	168
6	218	417	324	72	46	500	521
11	187	146	68	72	11	17	82
12	391	289	8	72	46	92	93
13	194	111	31	72	46	173	219
14	194	111	31	72	46	173	219
17	216	144	67	72	43	145	150
19	210	466	365	72	44	97	100
20	416	136	12	72	44	181	132
23	234	105	28	72	46	1	175
27	916	30	7	72	46	50	68
29	172	117	50	0	35	185	189

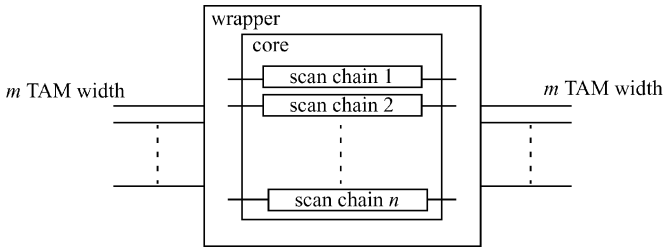


Fig. 2. Scan chains design at a core.

chain is connected to a TAM wire. For instance, if the n scan chains in Fig. 2 are connected to m wrapper chains ($n \geq m$), the loading of a new test vector can be performed in the m wrapper chains concurrently.

The problem of forming wrapper chains has been addressed by Iyengar *et al.* [15] and the test application time for a core is given by:

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

where s_i is the longest wrapper chain for scan in, s_o is the longest wrapper chain for scan out, and p is the number of test vectors.

The computation of an exact test time requires an algorithm such as the one proposed by Iyengar *et al.* to determine the number of wrapper chains at a core. It is important to note that even if the test time is computed exactly for each testable unit, the above formula does not consider the effect at the system level introduced by the clocked bypass structures, which is used in the TestShell when the TAM wires are becoming longer. Since the application of the formula at the system level does not lead to exact testing time and in order to reduce the computational cost, we suggest an approximation of the test time

$$\tau_a = \left\lceil \frac{\tau_{e1}}{m} \right\rceil$$

where τ_{e1} is the test time when a single wrapper chain is assumed and m is the number of TAM wires assigned to the core, and m is in the designer specified range.

We have analyzed the correlation between the exact test-time (τ_e) computation and the approximated test time (τ_a). We have used one of the largest industrial designs, the P93791, in the ITC'02 benchmarks [35] to illustrate our analysis. We have extracted the twelve scan-based cores in the P93791. Key data for the design is in Table I. The aim of the analysis is to check the correlation between the exact computation of the test time and our proposed approximation of the test time, and also to identify possible reasons for a nonlinear dependency between test time and the number of wrapper chains.

The analysis results for cores 1, 6 and 11 are collected in Table II, the results for cores 12–14 in Table III, the results for cores 17, 19 and 20 in Table IV, and the results for cores 20, 23, and 27 in Table V. For each core, we have computed the exact test time using the wrapper chain partitioning algorithm presented by Iyengar *et al.* [15] for different cases of wrapper chains (TAM width) 1 to 16. For each width, we have also computed the approximate test time (τ_a) and the difference between the exact test time and the approximated test time. From the results in Tables II–V, we observe that the difference between τ_e and τ_a is extremely low for low TAM width. However, as the TAM width increases, the difference between τ_e and τ_a also increases. Among the cores, the case is worst for core 11. We have, therefore, made an investigation of core 11, and we made two observations. The number of scan chains is 11 while the TAM bandwidth has been in the range from 1 to 16, and the length of the scan chains is rather unbalanced. The shortest scan chain is 17 flip-flops long while the longest scan chain consists of 82 flip-flops. We have made three new scan chain partitions of core 11 (Table VI), namely:

- *balanced.11*—where the 11 scan chains are redesigned to be as balanced as possible;
- *balanced.22*—where the number of scan chains is increased to 22 and the partitions are made as balanced as possible;
- *balanced.44*—where the number of scan chains is increased to 44 and the partitions are made as balanced as possible;

TABLE II
TEST-TIME COMPARISON BETWEEN AN EXACT METHOD AND AN APPROXIMATION FOR CORES 1, 6, AND 11 IN P93791

	Core 1			Core 6			Core 11		
TAM width	**Test time, τ_e**	**Test time, τ_a**	**Diff (%) $\lvert\tau_e-\tau_a\rvert/\tau_e\times 100$**	**Test time, τ_e**	**Test time, τ_a**	**Diff (%) $\lvert\tau_e-\tau_a\rvert/\tau_e\times 100$**	**Test time, τ_e**	**Test time, τ_a**	**Diff (%) $\lvert\tau_e-\tau_a\rvert/\tau_e\times 100$**
1	2862952	2862952	0%	5317007	5317007	0%	149381	149381	0%
2	1431714	1431476	0.02%	2658613	2658504	0.004%	74784	74691	0.12%
3	954862	954317	0.06%	1809815	1772336	2.1%	49981	49794	0.4%
4	740459	715738	3.3%	1358456	1329252	2.1%	37580	37345	0.6%
5	573163	572590	0.1%	1126316	1063401	5.6%	32513	29876	8.1%
6	494049	477159	3.4%	907097	886168	2.3%	25177	24897	1.1%
7	431729	408993	5.3%	793217	759572	4.2%	21608	21340	1.2%
8	370639	357869	3.4%	679337	664626	2.2%	18977	18673	1.6%
9	318561	318106	0.14%	674957	590779	12.5%	17105	16598	3.0%
10	308319	286295	7.1%	565457	531701	6.0%	16538	14938	9.7%
11	305449	260268	14.8%	561077	483364	13.9%	15603	13580	13.0%
12	248049	238579	10.6%	455738	443084	2.8%	15603	12448	20.2%
13	246409	220227	10.6%	451577	409001	9.4%	15603	11491	26.4%
14	244359	204497	16.3%	451358	379786	15.9%	15603	10670	31.6%
15	191464	190863	0.3%	447197	354467	20.8%	15603	9959	36.2%
16	186549	178935	4.1%	341858	332313	2.8%	15603	9336	40.2%

TABLE III
TEST-TIME COMPARISON BETWEEN AN EXACT METHOD AND AN APPROXIMATION FOR CORES 12–14 IN P93791

	Core 12			Core 13			Core 14		
TAM width	**Test time, τ_e**	**Test time, τ_a**	**Diff (%) $\lvert\tau_e-\tau_a\rvert/\tau_e\times 100$**	**Test time, τ_e**	**Test time, τ_a**	**Diff (%) $\lvert\tau_e-\tau_a\rvert/\tau_e\times 100$**	**Test time, τ_e**	**Test time, τ_a**	**Diff (%) $\lvert\tau_e-\tau_a\rvert/\tau_e\times 100$**
1	1813502	1813502	0%	1893564	1893564	0%	1893564	1893564	0%
2	906947	906751	0.02%	946880	946782	0.01%	946880	946782	0.01%
3	604799	604501	0.05%	639989	631188	1.4%	639989	631188	1.4%
4	453892	453376	0.1%	480479	473391	1.5%	480479	473391	1.5%
5	363774	362700	0.3%	395654	378713	4.3%	395654	378713	4.3%
6	302596	302250	0.11%	320969	315594	1.7%	320969	315594	1.7%
7	259492	259072	0.16%	278654	270509	2.9%	278654	270509	2.9%
8	227337	226688	0.29%	242384	236696	2.3%	242384	236696	2.3%
9	217951	201500	7.5%	235754	210396	10.8%	235754	210396	10.8%
10	182278	181350	0.51%	200069	189356	5.4%	200069	189356	5.4%
11	181887	164864	9.4%	193439	172142	11.0%	193439	172142	11.0%
12	151689	151125	0.4%	165749	157797	4.8%	165749	157797	4.8%
13	145823	139500	4.3%	159509	145659	8.7%	159509	145659	8.7%
14	145431	129536	10.9%	153269	135255	11.8%	153269	135255	11.8%
15	145431	120900	16.9%	152879	126238	17.4%	152879	126238	17.4%
16	114060	113344	0.6%	123434	118348	4.1%	123434	118348	4.1%

- *balanced.88*—where the number of scan chains is increased to 88 and the partitions are made as balanced as possible.

The results from the experiments on the original core 11 and the four versions of the balanced design are collected in Table VII. We made experiments with the TAM width in the range from 1 to 16, and for each of the versions of core 11, at each TAM width, we computed the exact test time, the approximated test time and the difference between the exact time and the approximated time. On average, the approximated test time is 12.1% from the exact test time on the original design. For the balanced.11, which is the balanced version of the original one, the average is down to 5.7%. If the number of scan chains are increased to 22 as in balanced.22 the average difference is only 2.7% and if the number of scan chains are increased to 44 the average difference is down to 1.5%. A further increase of the

TABLE IV
TEST-TIME COMPARISON BETWEEN AN EXACT METHOD AND AN APPROXIMATION FOR CORES 17, 19, AND 20 IN P93791

	Core 17			Core 19			Core 20		
TAM width	**Test time, τ_e**	**Test time, τ_a**	**Diff (%) $\lvert\tau_e-\tau_a\rvert/\tau_e\times 100$**	**Test time, τ_e**	**Test time, τ_a**	**Diff (%) $\lvert\tau_e-\tau_a\rvert/\tau_e\times 100$**	**Test time, τ_e**	**Test time, τ_a**	**Diff (%) $\lvert\tau_e-\tau_a\rvert/\tau_e\times 100$**
1	1433858	1433858	0%	1031266	1031266	0%	3193678	3193678	0
2	717181	716929	0.04%	515843	515633	0.04%	1597047	1596839	0.01%
3	483258	477953	1.1%	343904	343755	0.04%	1065003	1064559	0.04%
4	358699	358465	0.07%	258027	257816	0.08%	798940	798419	0.07%
5	290128	286772	1.2%	206553	206253	0.15%	639256	638736	0.08%
6	257361	238976	7.1%	179524	171878	4.3%	551690	532280	3.5%
7	225028	204837	9.0%	156728	147324	6.0%	477047	456240	4.4%
8	192912	179232	7.1%	134801	128908	4.4%	416582	399210	4.2%
9	161664	159318	1.5%	114775	114585	0.17%	361121	354853	1.7%
10	160579	143386	10.7%	111164	103127	7.2%	346109	319368	7.7%
11	130849	130350	0.4%	94095	93751	0.4%	291064	290334	0.25%
12	129331	119488	7.6%	90077	85939	4.6%	286061	266140	7.0%
13	128680	110297	14.3%	85444	79328	7.2%	271466	245668	9.5%
14	128029	102418	20.0%	83133	73662	11.4%	261458	228120	12.8%
15	97215	95591	1.7%	68992	68751	0.35%	216005	212912	1.4%
16	97215	89616	7.8%	67506	64454	4.5%	215588	199605	7.4%

TABLE V
TEST-TIME COMPARISON BETWEEN AN EXACT METHOD AND AN APPROXIMATION FOR CORES 23, 27, AND 29 IN P93791

	Core 23			Core 27			Core 29		
TAM width	**Test time, τ_e**	**Test time, τ_a**	**Diff (%) $\lvert\tau_e-\tau_a\rvert/\tau_e\times 100$**	**Test time, τ_e**	**Test time, τ_a**	**Diff (%) $\lvert\tau_e-\tau_a\rvert/\tau_e\times 100$**	**Test time, τ_e**	**Test time, τ_a**	**Diff (%) $\lvert\tau_e-\tau_a\rvert/\tau_e\times 100$**
1	1836917	1836917	0%	2869269	2869269	0%	1161619	1161619	0%
2	918609	918459	0.02%	1435093	1434635	0.03%	584201	580810	0.6%
3	612618	612306	0.05%	957346	956423	0.1%	389582	387206	0.6%
4	475404	459229	3.4%	718007	717317	0.1%	292187	290405	0.6%
5	367758	367383	0.1%	588713	573854	2.5%	232497	232324	0.07%
6	317719	306153	3.6%	480507	478212	0.5%	194963	193603	0.7%
7	277534	262417	5.4%	418151	409896	2.0%	166241	165946	0.2%
8	240874	229615	4.7%	365882	358659	2.0%	161235	145202	9.9%
9	204440	204102	0.16%	341123	318808	6.5%	130090	129069	0.8%
10	200689	183692	8.5%	303526	286927	5.5%	129057	116162	10.0%
11	194579	166992	14.2%	279684	260843	6.7%	128884	105602	18.1%
12	160739	153076	4.8%	248506	239106	3.8%	97568	96802	0.8%
13	160504	141301	12.0%	248506	220713	11.2%	96879	89355	7.8%
14	156979	131208	16.4%	232000	204948	11.7%	96879	82973	14.4%
15	122898	122461	0.35%	217328	191285	12.0%	96706	77441	19.9%
16	120554	114807	4.8%	187067	179329	4.1%	96533	72601	24.8%

number of scan chains to 88, balanced.88, will not give a lower difference at the TAM bandwidth we did experiments with. The analysis indicates that designing the scan chains at a core in a balanced way with a relatively high number of scan chains will result in a near linear dependency between test time and TAM width. It should be noted that we used TAM bandwidth in the range from 1 to 16. Obviously, the linear dependency does not hold for small cores with a few scanned element. However, our modeling assumes a linear dependency within a range specified by the designer.

B. Test-Power Consumption

The power consumption is usually higher during testing compared to that during normal operation. The reason is that a high switching activity is desired in order to detect as many faults as possible for each test vector. Detecting a high number of faults

TABLE VI
MODIFIED SCAN-CHAIN PARTITIONING ON CORE 11

	# scan chains	Scan chains
Original	11	82 82 82 81 81 81 18 18 17 17 17
Balanced.11	11	53 53 53 53 52 52 52 52 52 52 52
Balanced.22	22	27 27 27 27 26 26 26 26 26 26 26 26 26 26 26 26 26 26 26 26 26 26
Balanced.44	44	14 14 14 14 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13
Balanced.88	88	7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6

TABLE VII
TEST-TIME COMPARISON BETWEEN AN EXACT MODEL (τ_e) AND OUR APPROXIMATION (τ_a) FOR VARIATIONS OF CORE 11 (SEE TABLE VI)

TAM width	Original			Balanced.11		Balanced.22		Balanced.44		Balanced.88	
	τ_a	τ_e	Diff (%) $\lvert \tau_e - \tau_a \rvert / \tau_e \times 100$	τ_e	Diff (%) $\lvert \tau_e - \tau_a \rvert / \tau_e \times 100$	τ_e	Diff(%) $\lvert \tau_e - \tau_a \rvert / \tau_e \times 100$	τ_e	Diff(%) $\lvert \tau_e - \tau_a \rvert / \tau_e \times 100$	τ_e	Diff(%) $\lvert \tau_e - \tau_a \rvert / \tau_e \times 100$
1	149381	149381	0	149381	0	149381	0	149381	0	149381	0
2	74691	74784	0.1	74784	0.1	74784	0.1	74784	0.1	74784	0.1
3	49794	49981	0.4	49981	0.4	49981	0.4	49981	0.4	49981	0.4
4	37345	37580	0.6	37579	0.6	37579	0.6	37579	0.6	37579	0.6
5	29876	32513	8.1	32135	7.0	30065	0.6	30064	0.6	30064	0.6
6	24897	25177	1.1	25177	1.1	25177	1.1	25177	1.1	25177	1.1
7	21340	21608	1.2	21806	2.1	21618	1.3	21608	1.2	21608	1.2
8	18673	18977	1.6	20487	8.9	18976	1.6	18976	1.6	18976	1.6
9	16598	17105	3.0	19739	15.9	16905	1.8	16909	1.8	16909	1.8
10	14938	16538	9.7	19739	24.3	16161	7.6	15219	1.8	15219	1.8
11	13580	15603	13.0	13903	2.3	13903	2.3	13903	2.3	13903	2.3
12	12448	15603	20.2	12775	2.6	12775	2.6	12775	2.6	12775	2.6
13	11491	15603	26.4	11839	2.9	11831	2.9	11835	2.9	11836	2.9
14	10670	15603	31.6	11090	3.8	10903	2.1	10903	2.1	10898	2.1
15	9959	15603	36.2	11086	10.2	10899	8.6	10147	1.9	10147	1.9
16	9336	15603	40.2	10338	9.7	10337	9.7	9582	2.6	9582	2.6
	Average:		12.1		5.7		2.7		1.5		1.5

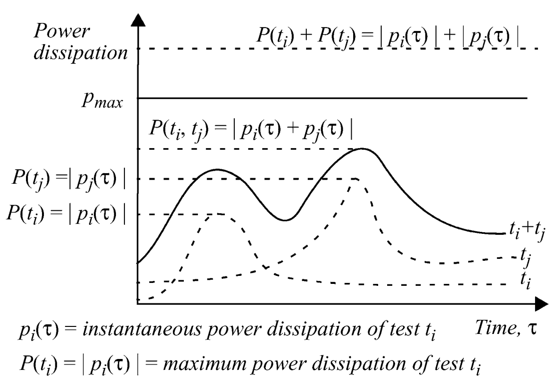


Fig. 3. Power dissipation as a function of time [5].

per vector minimizes the number of test vectors and therefore also the test time. The problem is that a high power consumption might damage the system. It is therefore needed to schedule the tests in such a way that the total test-power consumption is kept under control. To be able to analyze the test-power consumption, a model of the power consumption is needed. Chou *et al.* [5] have introduced a test-power model by denoting a fixed test-power value to each test set. The motivation for the model is as follows. Fig. 3 shows an example of the test-power dissipation variation over time τ when the two tests t_i and t_j are executed. If $p_i(\tau)$ and $p_j(\tau)$ are the instantaneous power dissipation of the two compatible tests t_i and t_j, respectively, and $P(t_i)$ and $P(t_j)$ are the corresponding maximal power dissipation. In the case where $p_i(\tau) + p_j(\tau) < p_{\max}$, the two tests can be scheduled at the same time. However, instantaneous power of each test vector is hard to obtain. To simplify the analysis, a fixed value $p_{\text{test}}(t_i)$ is assigned for all test vectors in a test t_i such that when the test is performed the power dissipation is no more than $p_{\text{test}}(t_i)$ at any moment. The $p_{\text{test}}(t_i)$ can be assigned as the average power dissipation over all test vectors in t_i or as the maximum power dissipation over all test vectors in t_i. The former approach could be too optimistic, leading to an undesirable test schedule, which exceeds the test-power constraints. The latter could be too pessimistic; however, it guarantees that the power dissipation will always satisfy the constraints. Usually, in a test environment the difference between the average and the maximal power dissipation for each test is often small since the objective is to maximize the circuit activity so that it can be tested in the shortest possible time [5]. Therefore, the definition of power dissipation $p_{\text{test}}(t_i)$ for a test t_i is usually assigned to the maximal test-power dissipation ($P(t_i)$) when test t_i alone is applied to the device. This simplification was introduced by Chou *et al.* [5] and has been used by Zorian [44] and by Muresan *et al.* [39]. We will also use this assumption in our approach.

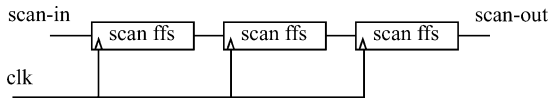


Fig. 4. Original scan chain [40].

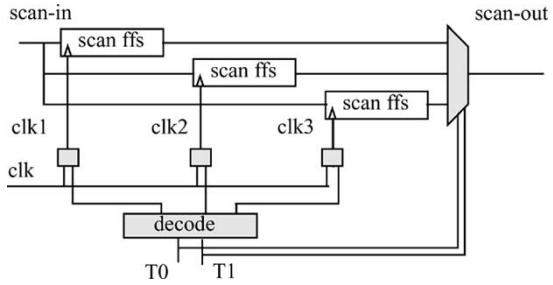


Fig. 5. Scan chain with gated subchains [40].

C. Test-Power Consumption at Test Parallelization

The test-power consumption depends on the switching activity. During testing in scan-based systems, switches appear not only during the application of test vectors, at the capture cycles, but also in the shift process when a new test vector is shifted in while the test response from the previous test vector is shifted out. Saxena *et al.* [40] proposed a gating scheme to reduce the test-power dissipation during the shift process. Given a set of scan chains as in Fig. 4 where the three scan chains are forming a single chain. During the shift process, all scan flip-flops are active and it leads to high switch activity in the system and high power consumption. However, if a gated subchain scheme as proposed by Saxena *et al.* is introduced (Fig. 5), only one of the three chains is active at a time during the shift process, while the others are switched off and as a result no switching activity is taking place in them. The test time in the examples (Figs. 4 and 5) are the same while the switch activity is reduced in the gated example and also the activity in the clock tree distribution is reduced [40]. The experimental results presented by Saxena *et al.* [40] indicate that the test-power consumption can be reduced to a third using a gated scheme with three sub chains as in Fig. 5 compared to the original scheme in Fig. 4. We use a generalized model based on the experimental results presented by Saxena *et al.*, which shows that there is a linear dependency between the test time and the test-power consumption. If x scan chains exist at a core, it is possible to form x number of wrapper chains. In such a case, the test time will be reduced since the shift process is minimized, however, the test-power consumption will be maximized since all of the x scan chains are active at the same time. On the other hand, if a single wrapper chain is assumed where all scan chains are connected into a single wrapper chain, the test time will increase but the test-power consumption can be reduced by gating the x scan chains. For the power modeling, we do as with the test time. We assign one test time value and one test-power consumption value at a single wrapper chain. As the number of assigned TAM wires changes, we assume that there is a linear change in test time and the test power within the specified range.

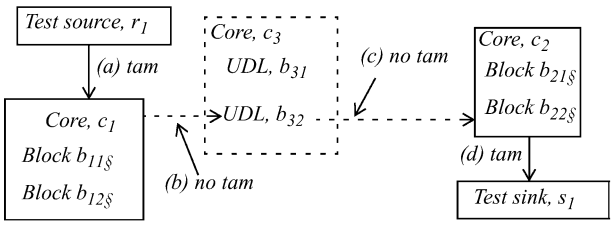


Fig. 6. Illustration of unwrapped core testing and test interference.

D. Test-Resource Limitations

A test source usually has a limited bandwidth. For instance, an external tester only supports a limited number of scan chains at a time [10]. The memory limitation at a test source may also put a limitation on the testing [10] and there could also be a limitation in the number of available pins. In our model, we use a fixed value for the maximal bandwidth at a test source and a test sink. We also use a fixed value to indicate the memory size used for test-vector storage at a test source.

E. Test Conflicts

We have discussed conflicts due to test-power consumption, bandwidth limitations, memory limitations, and sharing of TAM wires. These are conflicts that are to be considered during the test scheduling. There are also conflicts that are known in advance. In this section, we will discuss such test conflicts. These conflicts are due to interference during testing and also the testing of interconnections and UDL. It is also possible that conflicts appear when the system contains cores that are embedded in cores [19].

In general, in order to execute a test, a set of testable units might be required. This set can often be specified in advance. The advantage of specifying the test conflicts explicitly is that it gives the designer the flexibility to explore different design possibilities. It also makes our technique more flexible compared to an approach where test conflicts are built in to the tool.

We will use the example in Fig. 6 to illustrate unwrapped core testing and test interference. The example consists of only one test source and one test sink and three cores (c_1–c_3), where core c_1 consists of two testable units, b_{11} and b_{12}, core c_2 consists of two testable units, b_{21} and b_{22}, and core c_3 consists of two testable units, b_{31} and b_{32}. Cores c_1 and c_2 have interfaces to the TAM, i.e., they are placed in wrappers. As discussed above, we call such cores *wrapped cores* while cores such as core c_3, which do not have a dedicated interface to the TAM, are called *unwrapped cores*. Tests performed at wrapped cores are called *wrapped core tests*, and tests at unwrapped cores are called *unwrapped core test*. The testing of the UDL at the testable unit b_{32} is an unwrapped core test. In order to perform testing of b_{32}, the test vectors are transported from the test source r_1 using the TAM to the wrapped core c_1. The wrapper at c_1 is placed in the external test mode, which means that no core tests can be applied at c_1 as long as the wrapper is in external test mode. The test vectors are transported from the wrapper at c_1 to the testable unit b_{32}. The test responses are captured at the wrapper at core c_2, which, as core c_1, is placed in the external test mode. From the wrapper at core c_2, the test response is transported via the TAM to the test sink s_1. The testing of the testable units b_{11} and

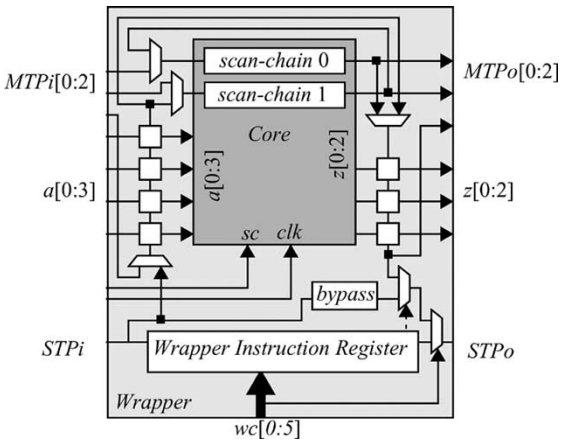


Fig. 7. Conceptual view of a P1500 compliant core [33].

b_{12} at core c_1 and b_{21} and b_{22} core c_2 cannot be performed at the same time as the testing of b_{32}. In our model we specify this as a conflict list: $\{b_{11}, b_{12}, b_{21}, b_{23}\}$.

The testing of b_{32} could, but does not have to, have an impact on the testing of some other testable units such as b_{31}. In our approach, we list all testable units that are required in order to execute a test explicitly. If b_{31} is interfered during the testing of b_{32}, b_{31} is also included in the conflict list.

An advantage of listing all the test conflicts explicitly is that it makes it possible to model hierarchy where for instance cores are embedded in cores. A hierarchical modeling technique usually has an implicit way to model conflicts. In our approach, such implicit modeling does not exist and, hence, longer conflict lists are required.

F. TAM Design

In our design flow, we assume that there are initially no TAM wires in the system. TAM wires are added when the transportation of test data in the system requests them. In general, all test sources are to be connected with wrapped cores and the wrapped cores have to be connected with the test sinks.

In our modeling of the TAM wires, we assume that each TAM wire is independent of other TAM wires. It means we are not partitioning the TAM wires into subsets. We also assume that the delay of data transportation on the TAM wires is negligible. Further, we assume, as discussed above, that the time impact from eventual bypass structures introduced on long time wires is not considered.

For the placement modeling of cores and test resources, we assume a single point assignment given by x- and y-coordinates for each. The placement model could be more elaborate than a single point assignment, however, a more advanced model would lead to higher computational complexity. To illustrate that further, consider a conceptional view of a P1500 compliant core given in Fig. 7. It is important to note that it is only a conceptional view. For instance, the inputs do not always have to be placed on the left hand side and the outputs on the right hand side. A more elaborate placement model than a single point model would need a way to determine where to connect the wires. A more elaborate wiring model would also have to consider the size of the core, since the wiring inside the wrapper has a cost.

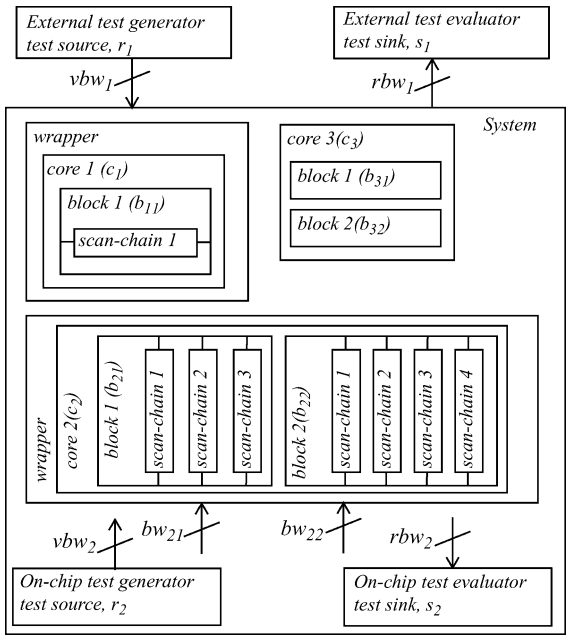


Fig. 8. Modeling the example system.

Furthermore, a more elaborate model should handle the wiring due to connecting the scan chains into a set of wrapper chains. It means that the model should consider exactly where on the core each of the scan inputs and each of the scan outputs are placed. On top of this, it is also needed to have a model for different types of cores. A core can be equipped with a built-in bypass structure or a special transparency mode, which can reduce the required wiring but will require consideration during the test time modeling since the transparency mode can require several clock cycles in order to transport test data from the core input to its outputs. There might also exist several possibilities for transparency where each such mode requires a certain number of clock cycles and each mode consumes a certain amount of power. We have, therefore, decided to use a single point model for the placement.

IV. System Modelling

In Section III, we discussed SOC test problems and their modeling. In this section, we describe our system model and the input specification to our test design tool. We illustrate the modeling and the input specification using an example.

We have developed a model, which is based on our previous system model [29], [30], [32], to model a SOC system with given test methods and added test resources, as illustrated in Fig. 8, a design with test (DT) is represented as follows.

DT $= (C, T, R_{\text{source}}, R_{\text{sink}}, p_{\max},)$, where: $C = \{c_1, c_2, \ldots, c_n\}$ is a finite set of cores; each core, $c_i \in C$, is characterized by:

(x_i, y_i): placement denoted by x and y coordinates and each core consists of a finite

set of blocks $c_i = \{b_{i1}, b_{i2}, \ldots, b_{im_i}\}$ where $m_i > 0\{i = 1,2,\ldots,n\}$. Each block, $b_{ij} \in B\{i = 1,2,\ldots,n, j = 1,2,\ldots,m_i\}$, is characterized by:
minbw_{ij}: minimal bandwidth,
maxbw_{ij}: maximal bandwidth, which are the minimal possible bandwidth and maximal possible bandwidth at a block, respectively.

Each block, b_{ij}, is attached with a finite set of tests, $b_{ij} = \{t_{ij1}, t_{ij2}, \ldots, t_{ijo_{ij}}\}$, where $o_{ij} > 0\{i = 1,2,\ldots,n, j = 1,2,\ldots,m_i\}$ and each test, $t_{ijk} \in T\{i = 1,2,\ldots,n, j = 1,2,\ldots,m_i, k = 1,2,\ldots,o_{ij}\}$, is characterized by
τ_{ijk}: test time (at TAM bandwidth 1),
p_{ijk}: test power [at TAM bandwidth 1 using a gated subchain scheme (discussed above)],
mem_{ijk}: required memory for test pattern storage.
cl_{ijk}: constraint list with blocks required for the test.

$R_{\mathrm{source}} = \{r_1, r_2, \ldots, r_p\}$ is a finite set of test sources where each test source, $r_i \in R_{\mathrm{source}}$, is characterized by:
(x_i, y_i): placement denoted by x and y coordinates,
vbw_i: vector bandwidth,
$vmem_i$: vector memory size.

$R_{\mathrm{sink}} = \{s_1, s_2, \ldots, s_q\}$ is a finite set of test sinks; where each test sink, $s_i \in R_{\mathrm{sink}}$, is characterized by:
(x_i, y_i): placement denoted by x and y coordinates,
rbw_i: response bandwidth,
source: $T \rightarrow R_{\mathrm{source}}$ defines the test sources for the tests;
sink: $T \rightarrow R_{\mathrm{sink}}$ defines the test sinks for the tests;
$p_{\max}$: maximal allowed power at any time.

The input specification for the example system (Fig. 8) to our test design tool is outlined in Fig. 9. The power limit for the system is given under [Global Constraints], and at [Cores], the placement (x, y) for each core is given and all blocks within each core are listed. The placement (x, y) for each test source, its possible bandwidth, and its test vector memory are given at [Generators]. At [Evaluators], the placement (x, y) and the maximal allowed bandwidth for each test sink is given. For each test the following is specified under [Tests], the test identifier, test power $\langle \mathrm{pwr} \rangle$, test time $\langle \mathrm{time} \rangle$, test source $\langle \mathrm{tpg} \rangle$, test sink $\langle \mathrm{tre} \rangle$, minimal $\langle \mathrm{minbw} \rangle$, and maximal bandwidth $\langle \mathrm{maxbw} \rangle$, memory requirement $\langle \mathrm{mem} \rangle$ and, optional, if the test is for testing of interconnections or UDL placed between another core $\langle \mathrm{ict} \rangle$. For instance, test t_2 is an *unwrapped core test* of UDL logic or interconnections between cores c_1 and c_2 (Fig. 9). Test t_2 requires at least two TAM wires but not more than four. The test source for test t_2 is r_1 and the test sink is s_2. The test vectors for test t_2 requires 5 units for storage at r_1. The tests for each block are specified at [Blocks] and at [Constraints] the blocks required in order to apply a test are listed. Note that the possibility to specify idle power for each block is implemented in our algorithm but, to simplify the discussion, it is excluded from the system model above.

The advantage with this model is that we can model a system with a wide range of tests (scan tests as well as nonscan tests such as delay, timing, and cross-talk tests) and constraints. For instance we can model:

- unwrapped core test, which is for the testing of interconnections and UDL;

```
# Example design
[Global Constraints]
MaxPower = 25
[Cores] identifier x   y    {block1, block2, ..., block n}
        c1   10   30   {b11}
        c2   10   20   {b21, b22}
        c3   20   30   {b31, b32}
[Generators] identifier x   y   maxbw   memory
        r1   10   40   3       100
        r2   10   10   1       100
[Evaluators] identifier x   y   maxbw
        s1   20   40   4
        s2   20   10   4
[Tests]  identifier  pwr time tpg  tre  minbw  maxbw  mem  ict
        t1          10  15   r1   s1   1      2      10   no
        t2          5   10   r1   s2   2      4      5    c2
    // for all tests in similar way
        t12     15  20   r1   s2   2      2      2    no
[Blocks] #Syntax: identifier idle pwr {test1, test2, ..., test n}
                b11    1      {t1, t2, t3}
                b21    1      {t4, t5}
    // for all blocks in similar way
                b32    2      {t11, t12}
[Constraints] Syntax: test {block1, block2,..., block n}
                  t1   {b11}
                  t2   {b11, b21, b22, b31, b32}
    // constraint for all tests in similar way
                  t12  {b11}
```

Fig. 9. Test specification of the system in Fig. 8.

- any combination of test resources, for instance a test source can be on-chip while the test sink is off-chip, and vice versa;
- any number of tests per block (testable unit);
- memory requirements at test sources;
- bandwidth limitations at test resources;
- constraints among blocks, which allows the modeling of constraints such as cores embedded in cores.

Initially, it is assumed that no TAM exists in the system. Our technique adds a set of TAMs, $\mathrm{tam}_1, \mathrm{tam}_2, \ldots, \mathrm{tam}_n$, where tam_i is a set of wires with a certain bandwidth. We assume that we can partition the TAM connected to a set of wrapped cores freely, which means we are not limited to assigning all wires in a TAM to one core at a time or dedicate TAM wires to a single core. We also assume that we can extend a subset of the TAM wires if required.

The TAM is modeled as a directed graph, $G = (N, A)$, where a node, $n_i \in N$, corresponds to a member of C, R_{source} or R_{sink}. An arc, $a_{ij} \in A$, between two nodes n_i and n_j indicates the existence of a wire and a wire w_k consists of a set of arcs. For instance, a wire w_1 from c_1 to c_3 passing c_2 is given by the two arcs: $\{a_{12}, a_{23}\}$.

The assignment of cores to TAM wires means connecting a test source n_{source} a set of cores $n_1, n_2, \ldots, n_n$ and a test sink n_{sink} and it is denoted as

$$n_{\mathrm{source}} \rightarrow [n_1, n_2 \ldots, n_n] \rightarrow n_{\mathrm{sink}} \tag{1}$$

where $[\,]$ indicates that these nodes (wrapped cores) are included (assigned) to this TAM but not ordered.

The length l_j of a test wire w_j is given by

$$\mathrm{dist}(n_i, n_j) + \sum_{k=2}^{n} \mathrm{adist}(n_{k-1}, n_k) + \mathrm{adist}(n_n, n_l) \tag{2}$$

where $n_i = n_{\text{source}}$ and $n_l = n_{\text{sink}}$ and the function adist gives the Manhattan distance between two nodes, i.e.:

$$\text{adist}(n_i, n_j) = |x_i - x_j| + |y_i - y_j|. \tag{3}$$

A set of wires form a tam_i and the routing cost is given by

$$\text{tamlength}_i = l_i \times \text{bandwidth}_i \tag{4}$$

where l_i is the length and bandwidth_i is the width of the TAM. The total TAM routing cost in the system is given by

$$c_{\text{tam}} = \sum_{\forall i} \text{tamlength}_i. \tag{5}$$

The total cost for a test solution is given by: $\alpha \times$ test time $+$ $\beta \times c_{\text{tam}}$, where test time is the total test application time, c_{tam} (defined above) is the total wiring cost, and α and β are two designer-specified constants determining the relative importance of the test time and the TAM cost.

V. Our Approach

In this section, we describe our approach to integrate test scheduling, test parallelization, and TAM design. For a given floor-planned system with tests, modeled as in Section IV, we have to:

- determine the start time for all tests,
- determine the bandwidth for each test,
- assign each test to TAM wires,
- determine the number of TAMs,
- determine the bandwidth of each TAM, and
- route each TAM

while minimizing the test time and the TAM cost, and considering constraints and power limitations. Note that when the start time and the bandwidth for a test are determined, the end time is implicitly given. Compared to previous approaches [29]–[32], we have the following improvements:

- *Test scheduling*. In [29] and [32], when a test was selected and all constraints were fulfilled, a TAM was designed. The technique there always minimized test time at the expense of the TAM cost. In the proposed approach, a cost function including both test time and TAM cost guides the test scheduling process.
- *Test parallelization*. The technique described in [31] maximized the bandwidth for each test, which resulted in low test time. However, its draw-back is a higher TAM cost. In our proposed approach, an elaborate cost function guides the algorithm.
- *TAM design*. In [29] and [32], when a test was considered and a free TAM existed, the TAM was selected. If an extension was required, an extension was made to minimize the additional TAM. A disadvantage of the approach is illustrated in Fig. 10, where *D* is to be connected using the dashed line [Fig. 10(a)]. A rerouting as *A, C, D, B* would include *D* at no additional cost [Fig. 10(b)].

The cost function for a test solution was defined above as: $\alpha \times$ test time $+ \beta \times c_{\text{tam}}$, where test time is the total test time, c_{tam} is the cost of the TAM and, α and β are user-defined constants used to determine the relative importance between the test time and the TAM cost. We also use the cost function for the guidance at each step in our algorithm based on each test and the TAM design. The cost function guiding our algorithm is for a test t_{ijk} using tam_l:

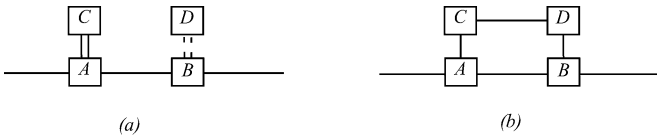


Fig. 10. Illustration of (a) a published and (b) our improved TAM design approach.

$$c = \alpha \times \tau_{\text{start}} + \beta \times \text{tamlength}_l \tag{6}$$

where: τ_{start}: is the time when t_{ijk} can start, tamlength_l: is the cost of the tam wiring (4), and the designer specified factors α and β are used to set the relative importance between test time and TAM cost. Equation (6) is used in the test scheduling algorithm for the selection of start time and TAM for each test.

A. Bandwidth Assignment

Test parallelization allows a flexible bandwidth assignment for each test depending on the bandwidth limitations at the block under test and the bandwidth limitations at the test resources.

The test time (see Section III-A) for a test t_{ijk} at block b_{ij} at core c_i is given by

$$\tau'_{ijk} = \left\lceil \frac{\tau_{ijk}}{bw_{ij}} \right\rceil \tag{7}$$

and the test power (see Section III-B)

$$p'_{ijk} = p_{ijk} \times bw_{ij} \tag{8}$$

where bw_{ij} is the bandwidth at block b_{ij} at core c_i [31].

Combining the TAM cost and the test time (7), we get for each block b_{ij} and its tests t_{ijk}

$$\text{cost}(b_{ij}) = \sum_{\forall\, k} l_l \times bw_{ij} \times \beta + \frac{\tau_{ijk}}{bw_{ij}} \times \alpha \tag{9}$$

where: $l_l = [\text{source}(t_{ijk}) \rightarrow c_i \rightarrow \text{sink}(t_{ijk})]$ and k is the index of all tests at the block. To find the minimum cost of (9), the derivative in respect to bw gives the bandwidth bw_{ij} at a block b_{ij}

$$\sqrt{\frac{\alpha}{\beta} \times \frac{\sum_{\forall\, k} \tau_{ijk}}{\sum_{\forall\, l} l_l}}. \tag{10}$$

Naturally, when selecting bw_{ij}, we also consider the bandwidth limitations at each block.

B. Test Scheduling

Our test-scheduling algorithm is outlined in Fig. 11. First, the bandwidth is determined for all blocks (Section V-A) and the tests are sorted based on a key (time, power or time $\times$ power). The outermost loop terminates when all tests are scheduled. In the inner loop, the first test is picked and after calling *create_tamplan* (Section V-C), the required number of TAM wires are selected or designed for the test based on the cost function. If the TAM factor is important, a test can be delayed in order to use an existing TAM. This is determined by the cost function. If all constraints are fulfilled, the test is

```
for all blocks bandwidth = bandwidth(block)
sort the tests descending based on time, power or time×power
τ=0
until all tests are scheduled begin
    until a test is scheduled begin
        tamplan = create_tamplan(τ, test) // see Figure 12 //
        τ′ = τ + delay(tamplan)
        if all constraints are fulfilled then
            schedule(τ′)
            execute(tam plan) // see Figure 13 //
            remove test from list
        end if
    end until
    τ = first time the next test can be scheduled
end until
order (tam) // see Figure 15 //
```

Fig. 11. Test-scheduling algorithm.

```
for all tams connecting the test source and test sink used by the test,
select the one with lowest total cost
  tam cost=0;
  demanded bandwidth=bandwidth(test)
  if bandwidth(test)>max bandwidth selected tam then
    demanded bandwidth=max bandwidth(tam)
    tam cost=tam cost+cost for increasing bandwith of tam;
  end if
  time=first free time(demanded bandwidth)
  sort tams ascending according to extension (τ, test)
  while more demanded bandwidth
    tam=next tam wire in this tam;
    tam cost=tam cost+cost(bus, demanded bandwidth)
    update demanded bandwidth accordingly;
  end while
  total cost=costfunction(tam cost, time, test);
```

Fig. 12. TAM estimation, i.e., $create\_tamplan(\tau, \mathrm{test})$.

scheduled and the TAM assignment is performed using the technique in Section V-C. Finally, all TAMs are optimized according to the technique discussed in Section V-E.

C. TAM Planning

In the TAM planning phase, our algorithm:

- creates the TAMs;
- determines the bandwidth of each TAM;
- assigns tests to the TAMs;
- determines the start time and the end time for each test.

The difference compared to the published approaches is that in the planning phase we only determine the existence of the TAMs but not their routing.

For a selected test, the cost function is used to evaluate all options ($create\_tamplan(\tau', \mathrm{test})$) (Fig. 12). The time ($\tau'$) when the test can be scheduled to start and its TAM is determined using the cost function and if all constraints are fulfilled, the TAM floor plan is determined [execute ($tamplan$)] (Fig. 13).

To compute the cost of extending a TAM wire with a node, the length of the required additional wires is computed. Since the order of the cores on a TAM is not decided, we need an

```
demanded bandwidth = bandwidth(test)
if bandwidth(test)>max bandwidth selected virtual tam then
   add a new tam with the exceeding bandwidth
   decrease demanded bandwidth accordingly
end if
time=first time the demanded bandwidth is free sufficient long
sort tams in the tam ascending on extension (test)
while more demanded bandwidth
   tam=next tam in this tam;
   use the tam by adding node(test) to it, and mark it busy
   update demanded bandwidth accordingly;
end while
```

Fig. 13. TAM modifications based on $create_tamplan$ (Fig. 12), i.e., execute ($tamplan$).

estimation technique for the wire length. For most TAMs, the largest wiring contribution comes from connecting the nodes with the largest distance from each other. The rest of the nodes can be added on the TAM at a limited additional cost (extra routing). However, for TAM's with a high number of nodes, the number of nodes becomes important.

Our estimation of the wire length considers both of these cases. We assume that the nodes (test sources, test sinks and the wrapped cores) in the system are evenly distributed over the area, i.e., $A = \text{width} \times \text{height} = (N_x \times \Delta) \times (N_y \times \Delta) = N_x \times N_y \times \Delta^2$, where N_x and N_y are the number of cores on the x and y axis, respectively. Therefore, Δ, the average distance between two nodes, is computed as:

$$\Delta = \sqrt{\frac{A}{N_x \times N_y}}. \tag{11}$$

The estimated length el_i of a wire w_i with k nodes is

$$el_i = \max_{1 \le j \le k} \{ l(n_{\text{source}} \rightarrow n_j \rightarrow N_{\text{sink}}), \Delta \times (k+1) \}. \tag{12}$$

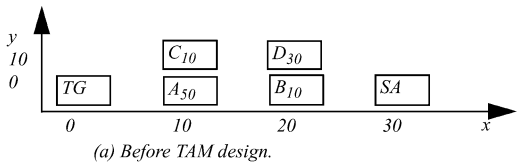
It means that we compute the maximum between the length of the longest created wire and the sum of the average distances for all needed arcs (wire parts). For example, let n_{furthest} be the node creating the longest wire, and n_{new} the node to be added, the estimated wiring length after inserting n_{new} is given by (12). (See equation at the bottom of the page.)

For a TAM, the extension is given as the summation of all extensions of the wires included in the TAM that are needed in order to achieve the required bandwidth. The TAM selection for a test t_{ijk} is based on the TAM with the lowest cost according to

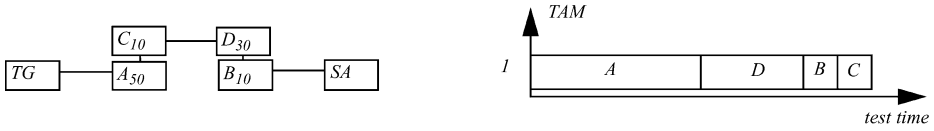
$$(el'_l - el_l) \times + \mathrm{delay}(\mathrm{tam}_l, \mathrm{t_{ijk}}) \times \alpha. \tag{13}$$

Using this cost function, we get a tradeoff between adding a new TAM and delaying a test on an existing TAM. For a newly created TAM, the delay for a test is 0 (since no other test is scheduled on the TAM and the test can start at time 0): $\mathrm{newcost}(t_{ijk}) = l(\mathrm{source}(t_j) \rightarrow c_i \rightarrow \mathrm{sink}(t_j)) \times \beta$.

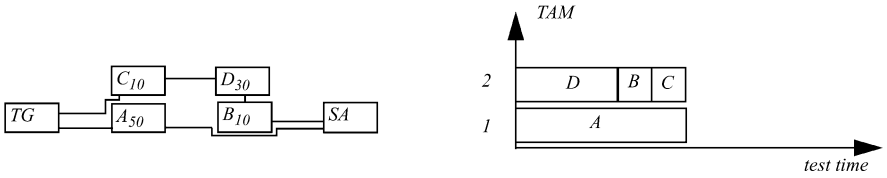
$$el'_i = \max \left\{ \min \left\{ \begin{matrix} l(n_{\text{source}} \rightarrow n_{\text{new}} \rightarrow n_{\text{furthest}} \rightarrow n_{\text{sink}}) \\ l(n_{\text{source}} \rightarrow n_{\text{furthest}} \rightarrow n_{\text{new}} \rightarrow n_{\text{sink}}) \end{matrix} \right\} \right\}. \\ \Delta \times (k+2)$$



(a) Before TAM design.



(b) TAM and corresponding test schedule with α:β factor 1:3.



(c) TAM and corresponding test schedule with α:β factor 2:3

Fig. 14. Example to illustrate TAM assignment.

TABLE VIII
ILLUSTRATION OF TAM ASSIGNMENT. ON TOP WITH A COST FUNCTION WHERE $\alpha = 1$ AND $\beta = 3$ AND BELOW WITH A COST FUNCTION, WHERE $\alpha = 2$ AND $\beta = 3$

Cost function	Step	Test/block	Length	TAM options	Cost	Schedule on selected TAM	Selected TAM
α=1 and β=3	1	A	50	New:TG→A→SA	30×3+0×1=90	0-50	New:TG→A→SA
	2	D	30	New:TG→D→SA T1:TG→[A,D]→SA	50×3+0×1=150 20×3+50×1=110	0-50-80	T1:TG→[A,D]→SA,
	3	B	10	New:TG→B→SA T1:TG→[A,D,B]→SA	30×3+0×1=90 0×3+80×1=80	0-50-80	T1:TG→[A,D,B]→SA
	4	C	10	New:TG→C→SA T1:TG→[A,D,B,C]→SA	50×3+0×1=150 0×3+90×1=80	0-50-80	T1:TG→[A,D,B,C]→SA
α=2 and β=3	1	A	50	New:TG→A→SA	30×3+0×2=90	0-50	T1:TG→A→SA
	2	D	30	New:TG→D→SA T1:TG→[A,D]→SA	50×3+0×2=150 20×3+50×2=160	0-30	T2:TG→D→SA
	3	B	10	New:TG→B→SA T1:TG→[A,B]→SA T2:TG→[D,B]→SA	30×3+0×2=90 0×3+50×2=100 0×3+30×2=60	0-30-40	T2:TG→[D,B]→SA
	4	C	10	New:TG→C→SA T1:TG→[A,C]→SA T2:TG→[D,B,C]→SA	50×3+0×2=150 20×3+50×2=110 0×3+40×2=80	0-30-40-50	T2:TG→[D,B,C]→SA

D. Example

We illustrate the TAM assignment by using an example with four cores each with one block (testable units). The four cores are placed as in Fig. 14(a). We assume that there is one test per block and that the test time is attached to each block. In the example, all tests are making use of the same test resources (test source and test sink) and there are no bandwidth limitations. Assuming an initial sorting of the tests based on test time, i.e., A, D, B, C, and success for the schedule at first attempt for each test (there are no limiting constraints which means that when a test is selected it can be scheduled) and the cost function $(\alpha : \beta)$

add test source and test sink to a final list
sort all cores descending according to Eq. 17
while cores left in the list
  remove first node from list and insert in the final list
  insert direct after the position where Eq. 15 is fulfilled
end while

Fig. 15. Routing optimization of all TAMs.

TABLE IX
ILLUSTRATION OF TAM ROUTING

Step	Selected	Order before selection	Length of each TAM partition	Order after selection	TAM length	Remaining in list
0		-		TG→SA	30	C,D,A,B
1	C	TG→SA	0	TG→C→S	14+22=36	D,A,B
2	D	TG→C→SA	18,2	TG→C→D→SA	14+10+14=38	A,B
3	A	TG→C→D→SA	6,14,20	TG→A→C→D→SA	10+10+10+14=44	B
4	B	TG→A→C→D→SA	20,14,14,6	TG→A→C→D→B→SA	10+10+10+10+10=50	-

is in Fig. 14(b) set to 1:3 and in Fig. 14(c) to 2:3. The TAM design algorithm is illustrated for the two cases in Table VIII and the results are in Fig. 14(b) and (c). In the case when $\alpha = 1$ and $\beta = 3$ (top Table VIII), at step 1, A is selected (first in the list). No TAM exists in the design and the cost for a new TAM to be created is 90, which comes from the distance connecting TG with A and A with SA ($TG \rightarrow [A] \rightarrow SA$), $10+10+10 = 30$, times the tam factor (β), which is 3. It is a new TAM, which means that there is no delay for the test; test A is scheduled at time 0 to 50. In the second step D is considered. The TAM created in step 1 can be extended or alternatively a new TAM can be created. Both options are estimated. The cost of a new TAM, $TG \rightarrow [A] \rightarrow SA$, is 150 while the cost of an extension of T1, $TG \rightarrow [A, D] \rightarrow SA$, is 110, computed as TAM extension 20×3 and delay on TAM 50×1. The delay on the TAM is due to that A occupies the TAM during 0 to 50. The algorithm selects to make use of the existing TAM. When all tests are assigned to TAMs the resulting TAM and test schedule are as in Fig. 14(b).

For Fig. 14(c), a different solution is created due to a different cost function ($\alpha = 2$ and $\beta = 3$) (see algorithm flow, bottom Table VIII). The example illustrates the importance of considering the test time and the TAM design in an integrated way. In Fig. 14(b), the result is a single TAM, which implies higher testing time while in Fig. 14(c) two TAMs are created, which makes it possible to reduce the testing time.

In the example, only a single TAM wire is assumed. In complex systems a higher number of TAM wires exists. In our approach, we handle that and we treat each wire independently, and when the question comes to select TAM wires for a test we explore all possibilities.

E. TAM Optimization

Above, we created the TAMs for the system, assigned each test to a TAM, determined the bandwidth of the TAMs and every test was given a start time and an end time in such a way that no conflicts and no limitations were violated. In this section, we discuss the routing of the TAMs, order (tam) in Fig. 11. Our approach is based on a simplification of an algorithm presented by Caseau and Laburthe [4]. The notation $TG \rightarrow [A, D] \rightarrow SA$ was above used to indicate that core A and D were assigned to the same TAM, however, the order of $[A, D]$ was not determined (1), which is the objective in this section. We use

$$n_{\text{source}} \rightarrow n_1 \rightarrow n_2 \cdots \rightarrow n_n \rightarrow n_{\text{sink}} \tag{14}$$

to denote that a TAM from n_{source} (the test source) to n_{sink} (the test sink) connects the cores in the order $n_{\text{source}}, n_1, n_2, \ldots, n_n, n_{\text{sink}}$.

TABLE X
TEST SCHEDULING RESULTS

Design	Approach	Test time	Difference to optimum (%)	CPU (s)
Muresan [39]	Optimal	25	-	-
	SA [30]	25	0	90
	Muresan [39]	29	16.0	-
	DATE [29]	26	4.0	1
	Our (p)	25	0	1
Kime [23]	Optimal	318	-	-
	Kime and Saluja[23]	349	9.7	-
	DATE [29]	318	0	1
	Our	318	0	1
System S [3]	Optimal	1152180		
	Chakrabarty SJF[2]	1204630	4.5	-
	DATE [29]	1152180	0	1
	Our	1152180	0	1
System L [29]	Optimal	1077	-	-
	DATE [29]	1077	0	1
	Our	1077	0	1
	Designer	1592	47.8	-
Ericsson [30]	Optimal	30899	-	-
	SA [30]	30899	0	3260
	DATE [29] (t)	34762	12.5%	3
	Our	30899	0	5

The TAM routing algorithm is outlined in Fig. 15. The algorithm is applied for each of the TAMs and, initially, in each case, the nodes (test sources, wrapped cores, and test sinks) of a TAM are sorted in descending order according to

$$\text{dist}(n_{\text{source}}, n_i) + \text{dist}(n_i, n_{\text{sink}}) \tag{15}$$

where the function dist gives the distance between two cores, or between a test source and a core, or between a core and a test sink, i.e.

$$\text{dist}(n_i, n_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}. \tag{16}$$

First, the test source and the test sink are connected (Fig. 15). In the loop over the list of nodes to be connected, each node is removed and added to the final list in such a way that the wiring distance is minimized according to

$$\min\{\text{dist}(n_i, n_{new}) + \text{dist}(n_{\text{new}}, n_{i+1}) - \text{dist}(n_i, n_{i+1})\} \tag{17}$$

where $1 \leq i < n$ (all nodes on the TAM).

We use the TAM, $TG \rightarrow [C, D, A, B] \rightarrow SA$, from the example in Fig. 14(a) to illustrate the algorithm, see Table IX.

TABLE XI
EXPERIMENTAL RESULTS ON INTEGRATED TEST SCHEDULING AND TAM DESIGN

Design	Approach	Test Application Time		Test Access Mechanism		Total		CPU
		Test time (τ)	Difference to SA (%)	TAM cost (tam)	Difference to SA (%)	Total cost α×τ+β×tam	Difference to SA(%)	
ASIC Z	SA [30]	326	-	180	-	506		865
	DATE[29]	262	-19.6	300	66.7	562	11.1	<1
	Our tp	262	-19.6	280	55.6	542	7.1	<1
	Our t	262	-19.6	280	55.6	542	7.1	<1
	Our p	305	-6.4	240	33.3	545	7.7	<1
Extended ASIC Z	SA [30]	270	-	560	-	830	-	4549
	DATE[29]	287	6.3	660	17.9	947	14.1	<1
	Our tp	264	-2.2	480	-14.3	744	-10.4	<1
	Our t	264	-2.2	460	-17.9	724	-12.8	<1
	Our p	264	-2.2	480	-14.3	744	-10.4	<1
System S	SA [30]	996194	0	160	-	1492194	-	1004s
	DATE[29] t	996194	0	320	100	1988194	33.2	<1
	Our tp	1152180	15.7%	100	-37.5	1462180	-2.0	<1
	Our t	1152180	15.7%	100	-37.5	1462180	-2.0	<1
	Our p	1152180	15.7%	100	-37.5	1462180	-2.0	<1
Ericsson	SA [30]	33082	-	6910	-	46902	-	15h
	DATE[29] tp	34762	5.1	8520	23.3	51802	10.4	62s
	Our tp	30899	-6.6	6015	-13.0	42929	-8.5	10s
	Our t	30899	-6.6	6265	-9.3	43429	-7.4	10s
	Our p	30899	-6.6	6205	-10.2	43309	-7.7	10s

At step 0, the nodes are ordered, C, D, A, B, and a connection is added between TG and SA. At step 1, in the loop over the sorted list, C is picked and inserted between TG and SA and the TAM is modified accordingly, $TG \rightarrow C \rightarrow SA$. Node C can obviously only be inserted in one way, that is between the test source and the test sink. At step 2, node D is to be inserted. D can be inserted as $TG \rightarrow D \rightarrow C \rightarrow SA$ or as $TG \rightarrow C \rightarrow D \rightarrow SA$. Equation (17) determines which of the alternatives to select and in this example $TG \rightarrow C \rightarrow D \rightarrow SA$ is selected. The algorithm continues until all nodes are inserted, resulting in a TAM: $TG \rightarrow A \rightarrow C \rightarrow D \rightarrow B \rightarrow SA$.

F. Complexity

The worst case complexity for the test scheduling when the TAM design is excluded is of $O(|T|^3)$ where T is the set of tests. In the TAM design there are two sequential steps; assignment and ordering (optimization). The assignment can be done in $O(|T| \times \log(|T|))$ and the optimization in $O(|n|^2)$ for a TAM with n cores. If we assume that each core consists of one block (testable unit) tested by one test set, the optimization is of $O(|T|^2)$. The test scheduling and the TAM assignment are integrated while the TAM optimization is performed as a final step. The total complexity is therefore $O(|T|^4 \times \log(|T|))$.

VI. EXPERIMENTAL RESULTS

We have implemented our technique and compared it with previously proposed approaches using the following designs: Ericsson [30], System L [29], System S [3], ASIC Z [44], an extended version of ASIC Z, a design called Kime [23] and one design named Muresan [39]. Detailed information about all benchmarks can be found at our web site [28].

TABLE XII
DESIGN ALTERNATIVES FOR THE ERICSSON DESIGN

Design alternative	α	β	test time	TAM cost
1	1	0	30899	19030
2	1	15	31973	5745
3	1	30	31973	5865
4	1	45	32901	5975
5	1	60	30899	5705
6	1	90	34332	5445
7	1	125	44488	5355
8	1	250	71765	4235
9	1	500	99016	4015
10	1	1000	134706	3635
11	0	1	235084	3175

When referring to our technique, unless stated, the reported results are produced based on an initial sorting of the test based on the key $t \times p$ (test time $\times$ test power), and our previous techniques are referred to as simulated annealing (SA) [30], [32] and DATE [29], [32]. The final cost and our algorithm are guided by the cost function: $\alpha \times$ test time $+ \beta \times$ TAM cost, where α and β are designer specified constants determining the relative im-

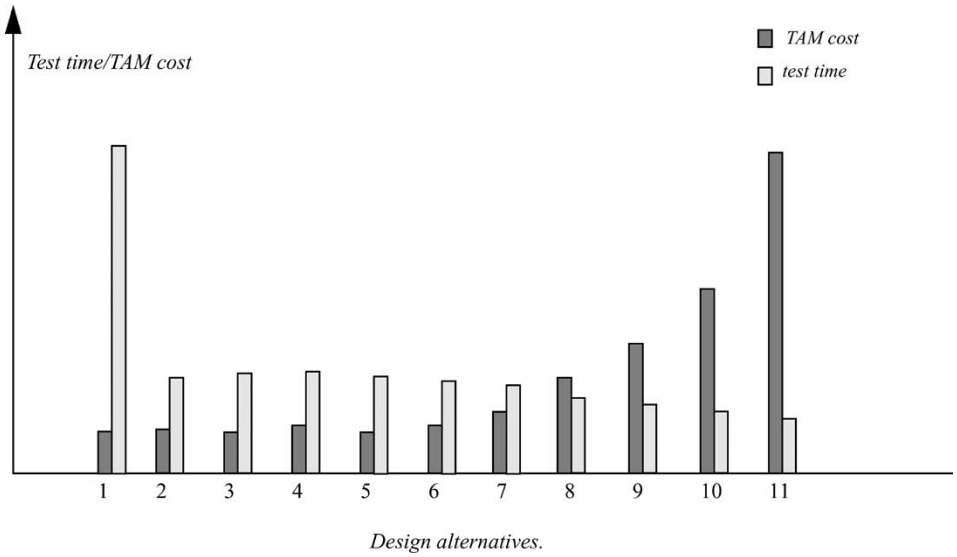


Fig. 16. Variation of test time and TAM cost for the design alternatives in Table XII of the Ericsson design.

TABLE XIII
EXPERIMENTAL RESULTS ON SYSTEM L ON COMBINED-TEST SCHEDULING, TAM DESIGN, AND TEST PARALLELIZATION

Approach	Test Application Time	Test Access Mechanism cost	CPU
SA [30]-flexible test time	316	N.A.	38s
DATE[29]-flexible test time	316	N.A.	<1
Our (α=15500:1)	316	18500	<1
Our (α=3000:β=1)	318	9490	<1
Our (α=500:β=1)	322	5140	<1
Our (α=100:β=1)	343	2420	<1
Our (α=50:β=1)	360	1750	<1
Our (α=10:β=1)	399	1030	<1
Our (α=5:β=1)	463	710	<1
Our (α=2:β=1)	593	510	<1
Our (α=1:β=1)	710	490	<1
Our (α=1:β=5)	923	380	<1
Our*	1077	240	<1

portance of the test time and the TAM cost (we use $\alpha = 1$ and $\beta = 1$ unless stated), see Section V.

For the experiments, we have used a Pentium II 350-MHz processor with 128-MB of RAM. Our previous results, referred to as DATE and SA, were performed on a Sun Ultra Sparc 10 with a 450-MHz processor and 256-MB of RAM [29], [30], [32].

A. Test Scheduling

We have performed experiments comparing our test scheduling technique with previously proposed techniques. The results are given in Table X. For design Muresan, the optimal test time can be computed to 25 time units. The approach proposed by Muresan *et al.* produces a solution with a test time of 29 time units. The DATE approach finds a solution using 26 time units at a computational cost of only 1 s. The SA optimization finds after 90 s the optimal solution while our technique finds it within 1 s. For the designs Kime, System S, and System L our approach finds the optimal solution at a low computational cost. For the larger Ericsson design, our approach finds the optimal solution after 5 s. The SA approach also finds the optimal solution, however, the computational cost is high. The DATE solution computes the solution after 3 s, but the solution is 12.5% from the optimal solution.

B. Integrated Test Scheduling and TAM Design

We have made experiments where we integrate the TAM and the test scheduling. For each of the benchmarks (ASIC Z, Extended ASIC Z, System S, and Ericsson), we applied our algorithm using the initial sorting of the tests based on the key, tp (time $\times$ power), t (time), and p (power). The results from the experiments are collected in Table XI. On ASIC Z when not considering idle power, the SA produces a solution with a test time of 326 time units and 180 as the TAM cost. The total cost of the solution is 506. The DATE approach produces a solution with a better test time than the SA approach, however, the TAM

cost is higher, leading to a higher total cost. Our approach produces solutions with a test time in the same range as the DATE approach. The TAM cost using our approach is better than the DATE approach leading to better total cost. The computational cost of our approach is in the same range as the DATE approach.

For the extended ASIC Z example, our approach produces test solutions with a lower test time in all cases compared to SA and the DATE approach. Furthermore, the TAM cost using our approach is lower than compared to the SA and the DATE approach, which leads to a lower total cost.

In the experiments on System S, the efficiency of our TAM design algorithm is shown. In the SA and DATE approaches the external tester supported several tests at the same time [30]. For our approach, we now assume that the external tester can support 2 tests concurrently, i.e., we have more limitation. The SA approach produced a solution with a TAM cost of 160 and the DATE produced a solution at a TAM cost of 320. Our approach results in a TAM cost of 100. The total cost is evaluated to 1 462 180 for our approach, which is to be compared to 1 492 194 using SA ($\alpha = 1$ and $\beta = 3100$).

For the experiments on the Ericsson design, we have used $\alpha = 1$ and $\beta = 2$ in the cost function. In the previous approaches, no bandwidth limitations were given on the external tester [29], [30], [32]. However, here we assume a bandwidth limitation of 12 (there are 12 wires or lines to distribute).

The experiments indicate that our approach produces test solutions at a low computational cost. The test time for the test solutions are in the range of the DATE solutions, however, the TAM costs are reduced leading to lower total costs. In many cases, our approach produces solutions near the SA solution at a low computational cost.

The advantage of a low computational cost is that it gives the designer a possibility to explore the design space since each iteration consumes only a low computational cost. We have computed the cost for a set of design alternatives for the Ericsson design (Table XII), which are plotted in Fig. 16. The results show that when test time decreases the TAM cost increases. Typically, the designer starts with the extreme points $\alpha = 0$, $\beta = 1$ (only TAM design is important) and $\alpha = 1$, $\beta = 0$ (only test time is important). And then creating new solutions at different values of α and β. The designer tries to find an appropriate balance of α and β based on the previous runs and inspection of the computed testing times and the TAM costs.

C. Test Scheduling, Test Parallelization, and TAM Design

In the experiments above, we assumed a fixed test time for each test set. In this experiment, we allow modifications of the test time at each test set. It means we have performed experiments combining test scheduling, TAM design, and test parallelization using the System L design. The results are collected in Table XIII. The results using SA and DATE approaches do not support TAM of higher bandwidth than 1 and therefore only test time is reported (the experiments were performed ignoring TAM design). In Our*, we have forced the bandwidth to 1.

VII. Conclusion

Test-time minimization and efficient TAM design are becoming increasingly important due to the high amount of test data to be transported in an SOC design. In the process of developing an efficient test solution, both test time and TAM design must be considered simultaneously. A simple TAM leads to long test application time while a more extensive TAM would reduce the test time at the cost of more wiring. However, an extensive TAM might not automatically lead to lower testing times since test resource conflicts and power limitations might limit the solution. We have proposed an integrated technique for test scheduling, test parallelization (scan chain partitioning) and TAM design that minimizes test time and TAM cost while considering test conflicts and power limitations. With our approach it is possible to model a variety of tests as well as tests of wrapped cores, unwrapped cores and UDL. We have implemented the technique and performed several experiments to demonstrate the efficiency of our approach.

References

[1] J. Aerts and E. J. Marinissen, "Scan chain design for test time reduction in core-based ICs," in *Proc. IEEE Int. Test Conf.*, Washington, DC, Oct. 1998, pp. 448–457.

[2] K. Chakrabarty, "Test scheduling for core-based systems using mixed-integer linear programming," *IEEE Trans. Computer-Aided Design*, vol. 19, pp. 1163–1174, Oct. 2000.

[3] ——, "Test scheduling for core-based systems," in *Proc. IEEE/ACM Int. Conf. Computer-Aided Design*, San Jose, CA, Nov. 1999, pp. 391–394.

[4] Y. Caseau and F. Laburthe, "Heuristics for large constrained vehicle routing problems," *J. Heuristics*, vol. 5, no. 3, pp. 281–303, 1999.

[5] R. Chou, K. Saluja, and V. Agrawal, "Scheduling tests for VLSI systems under power constraints," *IEEE Trans. VLSI Syst.*, vol. 5, pp. 175–185, June 1997.

[6] E. Cota, L. Carro, M. Lubaszewski, and A. Orailoglu, "Test planning and design space exploration in a core-based environment," in *Proc. Design, Automation, Test Eur. Conf.*, Paris, France, Mar. 2001, pp. 478–485.

[7] G. L. Craig, C. R. Kime, and K. K. Saluja, "Test scheduling and control for VLSI built-in-self-test," *IEEE Trans. Comput.*, vol. 37, pp. 1099–1109, Sept. 1988.

[8] A. L. Crunch, *Design for Test*. Englewood Cliffs, NJ: Prentice Hall, 1999.

[9] P. Harrod, "Testing reusable IP-a case study," in *Proc. IEEE Int. Test Conf.*, Atlantic City, NJ, Sept. 1999, pp. 493–498.

[10] G. Hetherington, T. Fryars, N. Tamarapalli, M. Kassab, A. Hassan, and J. Rajski, "Logic BIST for large industrial designs: Real issues and case studies," in *Proc. IEEE Int. Test Conf.*, Atlantic City, NJ, Sept. 1999, pp. 358–367.

[11] H.-S. Hsu, J.-R. Huand, K.-L. Cheng, C.-W. Wu, C.-T. Huang, and C.-W. Wu, "Test scheduling and test access architecture optimization for system-on-Chip," in *Proc. IEEE Asian Test Symp.*, Tamuning, Guam, Nov. 2002, pp. 411–416.

[12] Y. Huang, W.-T. Cheng, C.-C. Tsai, N. Mukherjee, O. Samman, Y. Zaidan, and S. M. Reddy, "Resource allocation and test scheduling for concurrent test of core-based SOC design," in *Proc. IEEE Asian Test Symp.*, Kyoto, Japan, Nov. 2001, pp. 265–270.

[13] Y. Huang, S. M. Reddy, W.-T. Cheng, P. Reuter, N. Mukherjee, C.-C. Tsai, O. Samman, and Y. Zaidan, "Optimal core wrapper width selection and SOC test scheduling based on 3-D bin packing algorithm," in *Proc. IEEE Int. Test Conf.*, Baltimore, MD, Oct. 2002, pp. 74–82.

[14] IEEE P1500 Standard for Embedded Core Test [Online]. Available: http://grouper.ieee.org/groups/1500/.

[15] V. Iyengar, K. Chakrabarty, and E. J. Marinissen, "Test wrapper and test access mechanism co-optimization for system-on-Chip," *J. Electron. Testing: Theory andApplicat.*, pp. 213–230, Apr. 2002.

[16] ——, "Efficient wrapper/TAM co-optimization for large SOCs," in *Proc. Design Test Eur.*, Paris, France, Mar. 2002, pp. 491–498.

[17] ——, "On using rectangle packing for SOC wrapper/TAM co-optimization," in *Proc. IEEE VLSI Test Symp.*, Monterey, CA, Apr. 2002, pp. 253–258.

[18] V. Iyengar, S. K. Goel, E. J. Marinissen, and K. Chakrabarty, "Test resource optimization for multi-site testing of SOC's under ATE memory depth constraints," in *Proc. IEEE Int. Test Conf.*, Baltimore, MD, Oct. 2002, pp. 1159–1168.

[19] V. Iyengar, K. Chakrabarty, M. D. Krasniewski, and G. N. Kuma, "Design and optimization of multi-level TAM architectures for hierarchical SOCs," in *Proc. IEEE VLSI Test Symp.*, Napa Valley, CA, Apr. 2003, pp. 299–304.
[20] S. K. Goel and E. J. Marinissen, "Cluster-Based test architecture design for system-on-chip," in *Proc. IEEE VLSI Test Symp.*, Monterey, CA, Apr. 2002, pp. 259–264.
[21] S. K. Goel and E. J. Mariniseen, "A novel test time reduction algorithm for test architecture design for core-Based system chips," in *Formal Proc. IEEE Eur. Test Workshop*, Corfu, Greece, May 2002, pp. 7–12.
[22] S. K. Goel and E. J. Marinissen, "Effective and efficient test architecture design for SOCs," in *Proc. IEEE Int. Test Conf.*, Baltimore, MD, Oct. 2002, pp. 529–538.
[23] C. R. Kime and K. K. Saluja, "Test scheduling in testable VLSI circuits," in *Proc. Int. Symp. Fault-Tolerant Comput.*, 1982, pp. 406–412.
[24] S. Koranne, "On test scheduling for core-based. SOCs," in *Proc. Int. Conf. VLSI Design*, Bangalore, India, Jan. 2002, pp. 505–510.
[25] ——, "Design of reconfigurable access wrappers for embedded core based SOC test," in *Proc. IEEE Int. Symp. Quality Electron. Design*, San Jose, CA, Mar. 2002, pp. 106–111.
[26] S. Koranne and V. Iyengar, "On the use of k-tuples for SOC test schedule representation," in *Proc. Int. Test Conf.*, Baltimore, MD, Oct. 2002, pp. 539–548.
[27] E. Larsson and H. Fujiwara, "Power constrained preemptive TAM scheduling," in *Formal Proc. IEEE Eur. Test Workshop.*, Corfu, Greece, May 2002, pp. 119–126.
[28] E. Larsson, A. Larsson, and Z. Peng. Linkoping Univ. SOC Test Site. [Online]. Available: http://www.ida.liu.se/labs/eslab/soctest/.
[29] E. Larsson and Z. Peng, "An integrated system-on-chip test framework," in *Proc. Design, Automation, and Test Eur. Conf.*, Munchen, Germany, Mar. 2001, pp. 138–144.
[30] E. Larsson, Z. Peng, and G. Carlsson, "The design and optimization of SOC test solutions," in *Proc. IEEE/ACM Int. Conf. Computer-Aided Design.*, San Jose, CA, Nov. 2001, pp. 523–530.
[31] E. Larsson and Z. Peng, "Test scheduling and scan-chain division under power constraint," in *Proc. IEEE Asian Test Symp.*, Kyoto, Japan, Nov. 2001, pp. 259–264.
[32] ——, "An integrated framework for the design and optimization of SOC test solutions," *J. Electron. Testing: Theory Applicat., Spec. Issue Plug-and-Play Test Automation System-on-a-Chip*, vol. 18, no. 4, pp. 385–400, Aug. 2002.
[33] E. J. Marinissen *et al.*, "Toward a standard for embedded core test: An example," in *Proc. IEEE Int. Test Conf.*, Atlantic City, NJ, Sept. 1999, pp. 616–627.
[34] E. J. Marinissen, S. K. Goel, and M. Lousberg, "Wrapper design for embedded core test," in *Proc. IEEE Int. Test Conf.*, Atlantic City, NJ, Oct. 2000, pp. 911–920.
[35] E. J. Marinissen, V. Iyengar, and K. Chakrabarty. ITC'02 SOC Test Benchmarks Web Site. [Online]. Available: http://www.extra.philips.com/itc02socbench/.
[36] E. J. Marinissen, R. Kapur, and Y. Zorian, "On using IEEE P1500 SECT for test plug-n-play," in *Proc. IEEE Int. Test Conf.*, Atlantic City, NJ, Oct. 2000, pp. 770–777.
[37] E. J. Marinissen, Y. Zorian, R. Kapur, T. Taylor, and L. Whetsel, "Toward a standard for embedded core test: An example," in *Proc. IEEE Int. Test Conf.*, Atlantic City, NJ, Sept. 1999, pp. 616–627.
[38] S. Mourad and Y. Zorian, *Principles of Testing Electronic Systems*. New York: Wiley, 2000.
[39] V. Muresan, X. Wang, V. Muresan, and M. Vladutiu, "A comparison of classical scheduling approaches in power-constrained block-test scheduling," in *Proc. IEEE Int. Test Conf.*, Atlantic City, NJ, Oct. 2000, pp. 882–891.
[40] J. Saxena, K. M. Butler, and L. Whetsel, "An analysis of power reduction techniques in scan testing," in *Proc. IEEE Int. Test Conf.*, Baltimore, MD, Oct. 2001, pp. 670–677.
[41] M. Sugihara, H. Date, and H. Yasuura, "A test methodology for core-based system LSIs," *IEICE Trans. Fundamentals*, vol. E81-A, no. 12, pp. 2640–2645, Dec. 1998.
[42] P. Varma and S. Bhatia, "A structured test re-use methodology for core-based system chips," in *Proc. IEEE Int. Test Conf.*, Washington, DC, Oct. 1998, pp. 294–302.
[43] C.-W. Wang, J.-R. Huang, Y.-F. Lin, K.-L. Cheng, C.-T. Huang, and C.-W. Wu, "Test scheduling of BISTed memory cores for SOC," in *Proc. IEEE Asian Test Symp.*, Tamuning, Guam, Nov. 2002, pp. 356–361.
[44] Y. Zorian, "A distributed BIST control scheme for complex VLSI devices," in *Proc. IEEE VLSI Test Symp.*, Atlantic City, NJ, Apr. 1993, pp. 4–9.

Erik Larsson (S'99–M'01) received the M.Sc. and Ph.D. degrees from Linköpings Universitet, Linköpings, Sweden, in 1994, 2000, respectively.

He is an Assistant Professor in the Department of Computer and Information Science, Linköpings Universitet. From October 2001 to December 2002, he held a Postdoctoral position funded by the Japan Society for the Promotion of Science at the Computer Design and Test Laboratory, Nara Institute of Science and Technology (NAIST), Nara, Japan. His current research interests include the development of tools and design for testability methodologies to facilitate the testing of complex digital systems. The main focuses are on system-on-chip test scheduling and test infrastructure design.

Klas Arvidsson received the B.E. degree in computer engineering in 2002 from Linköpings Universitet, Linköpings, Sweden, where he is currently pursuing the M.E. degree in socware.

His main research interest are embedded system design and compiler techniques.

Hideo Fujiwara (S'70–M'74–SM'83–F'89) received the B.E., M.E., and Ph.D. degrees in electronic engineering from Osaka University, Osaka, Japan, in 1969, 1971, and 1974, respectively.

He was with Osaka University from 1974 to 1985, with Meiji University from 1985 to 1993, and then joined Nara Institute of Science and Technology, Nara, Japan, in 1993. In 1981, he was a Visiting Research Assistant Professor at the University of Waterloo, Ontario, Canada, and, in 1984, he was a Visiting Associate Professor at McGill University, Montreal, Canada. Presently, he is a Professor at the Graduate School of Information Science, Nara Institute of Science and Technology. His research interests are in logic design, digital systems design and test, VLSI CAD and fault-tolerant computing, including high-level/logic synthesis for testability, test synthesis, design for testability, BIST, test-pattern generation, parallel processing, and computational complexity. He is the author of *Logic Testing and Design for Testability* (Cambridge, MA: MIT Press, 1985).

Prof. Fujiwara is an Advisory Member of the *IEICE Transactions on Information and Systems* and an editor of the IEEE TRANSACTIONS ON COMPUTERS, the *Journal on Electronic Testing*, the *Journal Circuits, Systems and Computers*, *Journal on VLSI Design*, and others. He received the IECE Young Engineer Award in 1977, the IEEE Computer Society Certificate of Appreciation Award in 1991, 2000, and 2001, the Okawa Prize for Publication in 1994, the IEEE Computer Society Meritorious Service Award in 1996, and the IEEE Computer Society Outstanding Contribution Award in 2001. He is a Golden Core Member of the IEEE Computer Society, a Fellow of the Institute of Electronics, Information and Communication Engineers of Japan (IEICE), and a Member of the Information Processing Society of Japan.

Zebo Peng (M'91–SM'02) received the Ph.D. degree in computer science from Linköpings University, Linköpings, Sweden, in 1987.

He is a Professor of the Chair in Computer Systems, Director of the Embedded Systems Laboratory, and Chairman of the Division for Software and Systems at Linköpings University. His current research interests include design and test of embedded systems, electronic design automation, design for testability, hardware/software co-design, and real-time systems. He has published over 140 journal and conference papers in these areas and coauthored the book *System Synthesis with VHDL* (Norwell, MA: Kluwer, 1997).

Prof. Peng has served on the Program Committee of several technical conferences and workshops, including DATE, DDECS, DFT, ECS, ETW, ITSW, FDL, and MEMOCODE, and was the General Chair of the 6th IEEE European Test Workshop (ETW'01). He currently serves as Vice-Chair of the IEEE European Test Technology Technical Council (ETTTC). He was the co-recipient of two Best Paper Awards at the European Design Automation Conferences (EURO-DAC) in 1992 and 1994. He is a Member of the ACM.