Resource-constrained system-on-a-chip test: a survey

Q. Xu and N. Nicolici

Abstract: Manufacturing test is a key step in the implementation flow of modern integrated electronic products. It certifies the product quality, accelerates yield learning and influences the final cost of the device. With the ongoing shift towards the core-based system-on-a-chip (SOC) design paradigm, unique test challenges, such as test access and test reuse, are confronted. In addition, when addressing these new challenges, the SOC designers must consciously use the resources at hand, while keeping the testing time and volume of test data under control. Consequently, numerous test strategies and algorithms in test architecture design and optimisation, test scheduling and test resource partitioning have emerged to tackle the resource-constrained core-based SOC test. This paper presents a survey of the recent advances in this field.

1 Introduction

The continuous advancement in the semiconductor manufacturing technology facilitates the integration of tens or even hundreds of millions of transistors onto a silicon die [1]. In order to meet the shrinking product development schedules and leverage the existing design expertise, SOC development is based on the design reuse philosophy, where two parties are involved: core providers and system integrators. Core providers create libraries of pre-designed and pre-verified building blocks, known as embedded cores, virtual components, macros or intellectual-property (IP) blocks. System integrators put the SOC together by combining the available cores and their custom user-defined logic (UDL) [2]. Examples of cores include embedded processors, digital signal processors, network controllers, memories and peripherals. Cores can be soft, hard or firm. A soft core comes in the form of synthesizable hardware description language code and has the advantage of being able to be easily re-targeted to different technologies. The trade-off for this flexibility and portability is unpredictability in area, performance and power. Hard cores, on the contrary, come with physical layout information and hence are technology-dependent. Despite the lack of flexibility, they are optimised for timing or power and have well-known performance parameters. Firm cores provide a trade-off between soft and hard cores. They are gate-level netlists and hence they are more predictable than soft cores, although, their size, aspect ratio and pin location can be tuned according to the system integrator's needs.

Although the functionality of cores can be made "plug-and-play" similar to the traditional printed circuit board (PCB) design, the SOC paradigm brings forth unique test challenges. Unlike the case of PCB design, where each chip is tested before the board production, and hence can be assumed fault-free to the system integrator, the cores integrated in a SOC need to be tested after the manufacturing process of the device [3]. As a result, SOC test is far more complex than the conventional board test, especially when the core's implementation is treated as a black box. The SOC test can be divided into three parts: tests for each individual core, tests for the UDL and tests for interconnect logic and wiring. This "divide and conquer" test strategy fits well into the SOC implementation paradigm. In addition, according to the analysis given by Kapur and Williams [4], a higher test quality for each core is required to achieve acceptable overall quality of the SOC, when compared to the case that the core itself is a chip.

There are mainly two approaches for testing embedded cores: built-in self-test (BIST) and applying and observing pre-computed test sets to and from the core terminals. The use of BIST-ed cores simplifies the integration of test plans. By setting the core in the BIST mode during test, the system integrator does not need to worry about the test access and test data translation problems. The on-chip BIST engine automatically generates test stimuli, usually through a linear feedback shift register (LFSR), and observes the test responses, typically using a multiple input signature register (MISR) [5]. However, although BIST eases the test tasks for system integrators, there are several limitations: (*i*) most of the cores are not BIST-ready, i.e. they contain random pattern resistant faults; (*ii*) for large cores it requires a significant design effort to achieve an acceptable fault coverage (even with the deterministic BIST [6]); and (*iii*) it generally leads to more area and performance overhead when compared to other design for test (DFT) techniques.

The more common scenario is that core providers supply their cores with a set of pre-computed test vectors that need to be applied and observed via the cores' terminals. These test sets may contain scan vectors, functional vectors or ordered test sequences for non-scanned sequential circuits. Some test vectors may be generated manually (e.g. reuse of functional simulation patterns) while some may be generated by automatic test pattern generation (ATPG) tools. Since the core test sets are created under the assumption that

IEE Proceedings online no. 20045019

doi: 10.1049/ip-cdt:20045019

Paper first received 3rd May and in revised form 13th December 2004

The authors are with Computer-Aided Design and Test Research Group, Department of Electrical and Computer Engineering, McMaster University, 1280 Main St. W., Hamilton, ON L8S 4K1, Canada

E-mail: nicola@ece.mcmaster.ca

all core terminals are directly accessible, the system integrator must ensure that the logic surrounding the core allows the core tests to be applied. Therefore, *test access* is a major concern that needs to be addressed by the system integrator. In addition, although for the UDL and interconnects the system integrator has full knowledge of the internal structure, because the tests are applied in the presence of other cores, test access is also a major concern. Since the number of cores used in an SOC may increase at a rate higher than the growth in the number of I/O pins, it is essential to develop test access architectures that are modular and scalable and cognizant of the limitations in the resources at hand. More specifically, the resource constraints are mainly determined by the number of chip's I/O pins, the number of tester channels and the depth of the tester buffers, the on-chip routing area necessary for transferring test data and the on-chip DFT area required for test-related logic. All of the above will influence the cost of the manufacturing test.

Recently a vast body of research has endeavoured to provide a better understanding of this research area. Numerous test strategies and algorithms in SOC test architecture design and optimisation, test scheduling and test resource partitioning have been proposed in the literature in order to reduce test cost of the SOC. Most of these endeavours focus on individual problems, and clearly, a comprehensive survey of the recent advances in this area is necessary. Iyengar *et al.* [7] presented an overview of modular SOC test planning techniques that address the problems of test architecture design and optimisation and constrained test scheduling algorithms. Test resource partitioning, which may significantly affect the cost of the resource-constrained SOC test, however, was not covered. In addition, many advanced methods have been proposed afterwards. In this paper, we present a more comprehensive review of the research in this domain. We first review different test access methods in Section 2. Then in Section 3 we overview the relevant solutions available for the design and optimisation of resource-constrained test architectures. Section 4 surveys the various test resource partitioning techniques. Finally we point out the potential research directions in Section 5.

2 SOC test access

Zorian *et al.* [3] proposed a conceptual infrastructure for SOC test, as illustrated in Fig. 1. The basic elements of this SOC test infrastructure are:

- *Test source and sink.* The test source generates test stimuli and the test sink compares the actual test responses to the expected responses. The test source and sink can be off-chip automatic test equipment (ATE), on-chip BIST hardware, or even an embedded microprocessor [8]. It is possible to have several test sources and sinks at the same time. It is also possible that the source and the sink are not of the same type. For example, an embedded core's test source can be ATE, while its test sink is a MISR located on-chip.
- *Test access mechanism (TAM)* facilitates the transport of test stimuli from the source to the core-under-test (CUT) and of the test responses from the CUT to the sink.
- *Core test wrapper* connects the core terminals to the rest of the chip and to the TAM, which isolates the embedded core from its environment during test. It facilitates modular testing at the cost of an additional area and potential performance overhead.

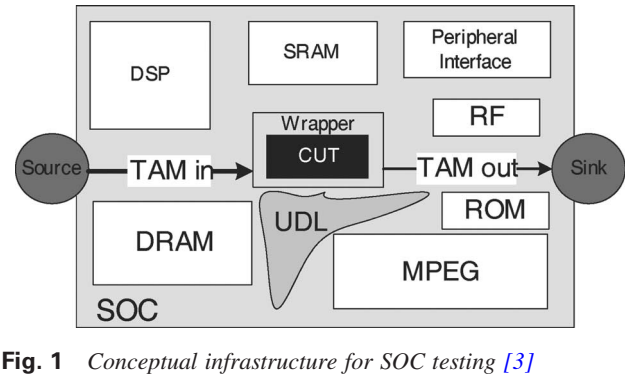


Fig. 1 *Conceptual infrastructure for SOC testing [3]*

One of the major challenges for resource-constrained SOC testing is to design an efficient TAM to link the test sources and sinks to the CUT [9]. There are a number of solutions for accessing the embedded cores from chip's I/O pins [10]: (*i*) direct parallel access via pin muxing; (*ii*) serial access and core isolation through a boundary scan-like architecture (also called isolation ring access mechanism); (*iii*) functional access through functional busses or transparency of embedded cores; and (*iv*) access through a combination of core wrappers and dedicated test busses. In this Section, we review the above test access strategies and compare their advantages and disadvantages from the cost standpoint.

2.1 Direct access

The simplest approach to test a core is to multiplex the core terminals to the chip level pins so that test patterns can be applied and observed directly [11]. The CUT can be tested as if it is the chip itself and hence the test sets/program can be reused almost without modification. It is also unnecessary to isolate the CUT using core test wrapper. While this approach apparently solves the TAM problem, due to the large number of cores and high number of core terminals, it introduces a large routing overhead. Furthermore, this approach does not scale well when the number of core terminals exceeds the number of SOC pins. In addition, since this approach does not provide access for the shadow logic of the CUT, it also results in degraded fault coverage.

Bhatia *et al.* [12] proposed a grid-based *CoreTest* methodology that uses a test-point matrix. Three types of test point (TP) were introduced: storage elements, embedded cells and observation test points. Direct parallel test access is provided through the "soft" netlist to these test points via the SOC I/O pins. The basic idea is to place bi-directional test points at the input and output of the embedded core and matrix accessible storage elements and observation test points in the UDL to provide sufficient fault coverage. This methodology allows the test logic and wiring used for testing UDL to be shared for testing cores and hence it reduces DFT area. However, new library cells for the proposed test points are required and global routing of the test grid is also necessary. In addition, test point selection and test matrix assignment require an additional development effort.

2.2 Isolation ring access

IEEE 1149.1 test architecture is a widely-used DFT technique to simplify the application of test patterns for testing interconnects at the board or system level [5]. A serial scan chain is built around every chip which allows indirect yet full access to all the I/O pins. In core-based SOC testing, system testability can be obtained in a similar way, by isolating each core using boundary scan and serially controlling and observing the I/Os of the core. This ring access mechanism can be implemented either internally

by the core provider or externally by the system integrator. In [13], Whetsel presented a test access architecture, which utilises the 1149.1 test access port (TAP) for core-level DFT and a novel TAP Linking Module for the overall SOC test control.

When compared to the direct access scheme, the isolation ring access has a significant lower routing overhead and supports testing cores with more ports than chip pins. Another important advantage lies in the effective core isolation from its surrounding logic, which not only protects the inputs of the CUT from external interference and the inputs of its superseding blocks from undesired values (e.g. bus conflicts), but it also facilitates the test of the surrounding logic by putting the core into the external test mode. However, the main shortcoming of this technique is the long test application time (TAT) due to the limited bandwidth caused by serial scan access. To decrease the hardware cost and performance overhead (i.e. a multiplexer delay for every path to and from the core) associated with full isolation ring, Touba and Pouya [14] described a method for designing partial isolation rings. This method avoids adding logic on critical paths, without affecting the fault coverage, by justifying a part of the test vectors through the surrounding UDL. Later in [15], the same authors proposed to modify the output space of the UDL and completely eliminate the need for an isolation ring for certain cores. Instead of scanning core test vectors into an isolation ring, the core test vectors are justified by controlling the inputs of the UDL. Despite avoiding the high number of multiplexers, the main limitation of the solution presented in [14, 15] lies in its computational complexity. This is because prior to deciding which input/output ring elements need to be inserted or removed, an analysis needs to be performed to check whether each test vector can be functionally justified. Therefore, the extensive usage of ATPG for this analysis reduces the reusability of the core tests and the scalability of the methodology.

2.3 Functional access

Functional access involves justifying and propagating test vectors through the available mission logic. Making use of the existing functional paths as test paths may significantly reduce the hardware cost. For a bus-based design, a large number of embedded cores are connected to the on-chip bus. Harrod [16] described a test strategy employed by ARM that enables test access through reusing the 32-bits functional bus. Test harness is introduced to isolate the AMBA-compliant core and communicate with the functional bus during test. In the test mode, the AMBA test interface controller becomes the AMBA bus master and is responsible for applying the test stimuli and capturing the test responses. The proposed approach is best suited for cores that are functionally tested and have a small number of non-bus I/Os. In addition, only one core is allowed to be tested at a time in this methodology, which may lead to long testing time.

Beenker *et al.* [17, 18] introduced a rather different test strategy, utilising the module transparency (e.g. existing functional paths) for test data transfer. Although this *Macro Test* was developed originally to improve test quality by testing "macros" with different circuit architectures (e.g. logic, memories, PLAs and register files) using different test strategies, later Marinissen and Lousberg [19] showed that the approach is also useful for testing core-based SOCs. The techniques described in Macro Test for introducing transparency, however, are rather *ad hoc*. Ghosh *et al.* [20] assumed that every core has a transparent mode in which data can be propagated from inputs to outputs in a fixed number of cycles. Their approach is based on the concept of finding functional test paths (F-paths) [21] through test control/data flow extraction. Although the proposed method successfully lowered the test area and delay overheads, it requires functional description of each core to make it transparent, which in most cases is not available or it is hard to extract. In addition, because test data cannot be pipelined through a core in this method, the potentially large transparency latency of each core (number of cycles needed to propagate test data from inputs to outputs of the core) may incur a relatively large test application time. To address the above issues, in [22] the same authors extended their approach by describing a trade-off between the silicon area consumed by the design-for-transparency hardware and propagation latency. They suggested that the core providers offer a catalogue of area/latency versions for their cores, from which the system integrator can select one in order to meet the test access needs of its neighboring cores. By associating a user-defined cost function with the transparent test paths in the core connectivity graph, the system integrator is able to select the version of cores that achieve an optimised test solution at the system level. This approach, however, requires a large design effort at the core provider side. Another limitation of [20, 22] stems from the difficulty in handling other popular DFT schemes, such as scan or BIST, in the SOC.

Solutions from [20, 22] require that all the I/Os of a core be simultaneously, though indirectly, controllable/observable. Ravi *et al.* [23] showed that this complete controllability and observability is unnecessary and proposed to provide them on an "as needed basis". Their approach allows for complex core transparency modes and is able to transfer test data to and from the CUT in a more aggressive and effective manner. For example, if the behaviour of a core is such that input data at times t_i and t_j combine to generate output data at time t_k, [23] can achieve the transparency objective without additional DFT cost, which is unlike [20, 22] that have to resort to the use of test multiplexers. This is achieved through transparency analysis using non-deterministic finite-state automata and transparency enhancement via symbolic justification and propagation based on a regular expression framework. Since integrated system synthesis with testability consideration is able to output a more efficient architecture compared to performing test modifications post-synthesis, later in [24] Ravi and Jha combined the synthesis flow of MOCSYN [25] with the symbolic testability analysis in [23] so that testability costs are considered along with other design parameters during synthesis.

The ability to apply an arbitrary test sequence to a core's inputs and observe its response sequence from the core's outputs consecutively at the speed of the system clock is important for non-scanned sequential circuits. It also facilitates an increase in the coverage of non-modelled and performance-related defects. Yoneda and Fujiwara [26] introduced the concept of consecutive transparency of cores, which guarantees consecutive propagation of arbitrary test stimuli/responses sequences from the core's inputs to the core's outputs with a propagation latency. An earlier synthesis-for-transparency approach presented by Chakrabarty *et al.* [27] had tackled the same problem, i.e. it made cores single-cycle transparent by embedding multiplexers. When compared to [27], the area overhead for making cores consecutively transparent with some latency instead of single-cycle transparent is generally lower [26]. In order to further decrease the hardware cost, the same authors [28]

proposed an integer linear programming (ILP) technique to make soft cores consecutively transparent with as few bypass multiplexers as possible. This is achieved by efficiently exploiting the data path description. Later, in [29], an area and time co-optimisation methodology for the TAM design problem was given using an ILP technique. In [30], Nourani and Papachristou presented a similar technique to core transparency, in which cores are equipped with a bypass mode using multiplexers and registers. They modelled the system as a directed weighted graph, in which the accessibility of the core input and output ports is solved as a shortest path problem. While this approach eases the problem of finding paths from the SOC inputs to the CUT, it requires packetisation of test data (to match the bit width of input and output ports), and the help of serialisation/de-serialisation bit-matching circuits.

2.4 Dedicated bus-based access

Since time-to-market is the overriding goal of core-based design, the ease with which cores can be designed and tested is crucial. As a result, the increased size of the logic and routing resources consumed by dedicated test infrastructure is acceptable for large SOC designs.

Aerts and Marinissen [31] described three basic scan-based test access architectures for core-based SOCs, as shown in Fig. 2:

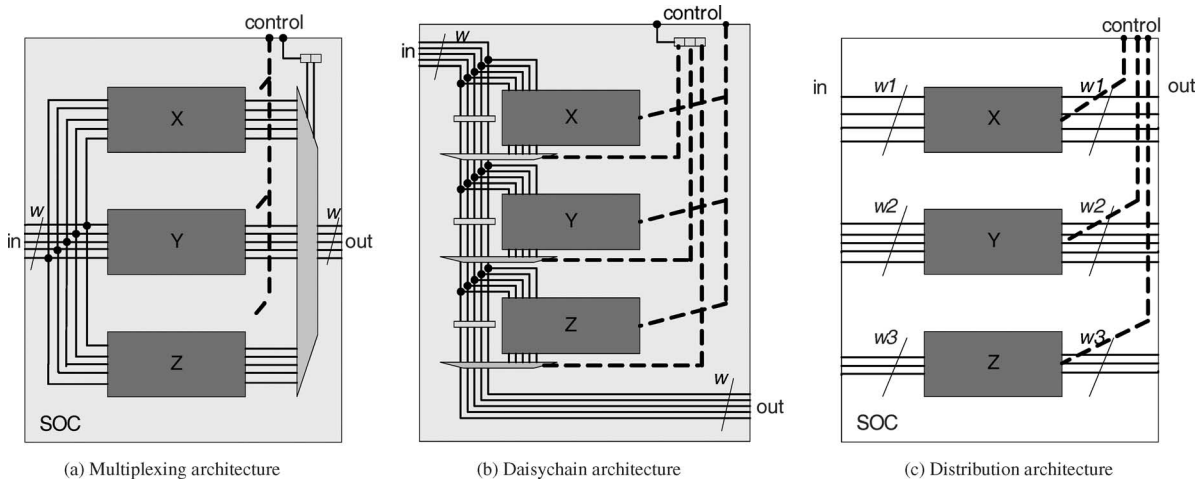
- *Multiplexing architecture:* All cores connect to all the TAM input pins and get assigned full TAM width. A multiplexer is added to select which core is actually connected to the TAM output pins, as shown in Fig. 2*a*. Only one core can be accessed at a time, and hence the total test application time is the sum of all the individual core test application times. Since core external testing needs to access two or more cores at the same time, it is hard to test circuits and interconnects between cores using this architecture (e.g. special wrapper design as described in [32] is required).
- *Daisychain architecture:* Long TAMs are constructed over all the cores from the TAM input pins to the TAM output pins. Bypasses are introduced to shorten the access paths to individual cores, as shown in Fig. 2*b*. Cores can be tested either simultaneously or sequentially in this architecture, however, concurrent testing increases test power with almost no test application time reduction. Due to the bypass mechanism, external testing can be done efficiently using this architecture.
- *Distribution architecture:* The total TAM width is distributed to all the cores, and each core gets assigned its private TAM lines, as shown in Fig. 2*c*. Hence the total TAM width has to be at least as large as the number of cores. Cores are tested concurrently in this architecture, and the test application time of the SOC is the maximum of individual core test application times. In order to minimise the SOC testing time, the width of an individual TAM should be proportional to the amount of test data that needs to be transported to and from a core connected to the TAM.

Based on the above basic architectures, two more popular test architectures, which support more flexible test schedules, were proposed. The *Test Bus* architecture proposed by Varma and Bhatia [33] can be seen as a combination of the Multiplexing and Distribution architectures, as shown in Fig. 3*a*. A single Test Bus is in essence the same as the Multiplexing architecture. Multiple Test Buses on an SOC operate independently (as in the Distribution architecture) and hence allow for concurrent testing, however, cores connected to the same Test Bus can only be tested sequentially. Cores connect the test bus through a proposed test wrapper called '*test collar*' in order to facilitate test access and test isolation. However, the test collar does not support test width adaptation and external testing of its surrounding logic. It should be noted that the width of the input and output test busses can be different in the original Test Bus architecture [33], although most of the later approaches consider them equal.

Marinissen *et al.* [34] proposed the *TestRail* architecture, which can be seen as a combination of the Daisychain and Distribution architectures, depicted in Fig. 3*b*. A single TestRail is in essence the same as the Daisychain architecture. Multiple TestRails on an SOC operate independently (as in the Distribution architecture). Cores on each TestRail can be tested sequentially as well as concurrently. A test wrapper called *TestShell* is proposed to connect each core to the TestRail. TestShell has four mandatory modes: function mode, IP test mode, interconnect test mode and bypass mode. A core specific test mode can be introduced, in addition to the four mandatory modes, if required. An advantage of the TestRail architecture over the Test Bus architecture is that it allows access to multiple or all wrappers simultaneously, which facilitates core external testing [35].

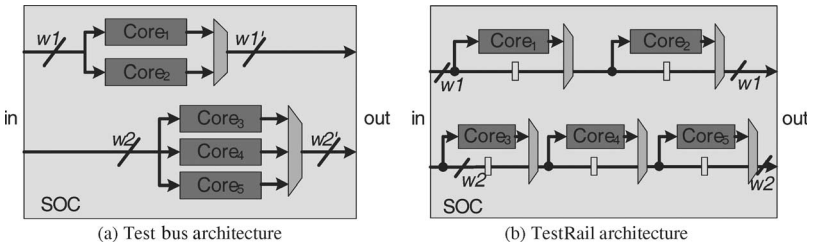
2.5 Test cost analysis

ITRS [1] anticipates that, if the current trends are maintained, the cost of testing a transistor will approach and may even exceed the cost of manufacturing it. Research in semiconductor testing in essence is the quest to find



(a) Multiplexing architecture (b) Daisychain architecture (c) Distribution architecture

Fig. 2 *Three basic test architectures [31]*



(a) Test bus architecture (b) TestRail architecture

Fig. 3 *Test bus and TestRail test architectures*

reliable yet low cost test strategies. In this Section we briefly discuss the cost of SOC test.

Nag *et al.* [36] pointed out that the cost of test (C_{test}) can be computed as follows $C_{test} = C_{prep} + C_{exec} + C_{silicon} + C_{quality}$. In the above formula C_{prep} stands for the fixed costs of test generation, tester program creation or any design effort for incorporating test-related features. C_{exec}, which is the cost of test execution, is mainly dependent on *automatic test equipment* (ATE) and $C_{silicon}$ stands for the cost required to incorporate DFT features on-chip. Finally, $C_{quality}$ accounts for the profit loss caused by performance degradation, test escapes and overkill.

To reduce C_{prep}, the test architecture must be optimised automatically in a short time, even for SOCs with a large number of cores. In addition, the translation of core-level tests to system-level tests should also be easily automated. $C_{silicon}$ is mainly dependent on the test access strategy and the amount of added test-related resources. To reduce $C_{quality}$, system integrator should try to decrease the performance overhead brought by DFT features and also balance fault coverage with yield loss. In particular, the system integrator must provide an efficient testing strategy for UDL and interconnects. To reduce C_{exec}, system integrators should avoid using high-end ATE and also minimise test application time by efficient test architecture optimisation and test scheduling techniques.

A generic comparison of the test cost for different test access strategies is shown in Table 1. The direct access scheme has the advantage of making core test translation unnecessary (low C_{prep}), however, it introduces a large routing overhead (high $C_{silicon}$) and has difficulty with testing UDL and interconnects (high $C_{quality}$). The main disadvantage of the ring access scheme is its long test application time (high C_{exec}). Although functional access through core transparency introduces the least DFT routing/area and performance overhead (low $C_{silicon}$ and $C_{quality}$), it is very difficult to design the system-level test access scheme (high C_{prep}). The modular test architecture using bus-based test access mechanism, being flexible and scalable, appears to be the most promising and cost-effective, especially for SOCs with a large number of cores. Its cost-effectiveness stems also from the fact that using design space exploration frameworks, the system integrator can trade-off different test cost parameter (e.g. C_{exec} against $C_{silicon}$). For bus-based TAMs, an acceptable $C_{quality}$ is guaranteed by providing test access to all the blocks in the SOC with the help of core test wrappers and the dedicated test busses. Within a satisfactory test development time (C_{prep}), the main source of SOC test cost reduction lies in the reduction of C_{exec} and $C_{silicon}$. The two most important factors in C_{exec} are the volume of ATE test data and the test application time. Test data volume affects both the number of scan clock cycles and the test data management on the ATE. For example, if test data volume exceeds the ATE buffer size, buffer reload is necessary, which takes longer than test application and hence dramatically increases the overall testing time. In addition, the test application time also depends on the test data bandwidth (the product of number of ATE channels and transfer rate), and more importantly the TAM distribution to cores and the test schedule. Therefore, the reduction in C_{exec} can be achieved through efficient test architecture optimisation and test scheduling techniques, which also affect the routing cost and the amount of DFT hardware, and hence $C_{silicon}$.

Table 1: Test cost comparison for different access strategies

Access strategy	C_{prep}	C_{exec}	$C_{silicon}$	$C_{quality}$
Direct access	Low	Medium	High	High
Isolation ring access	Medium	High	Medium	Medium
Transparency-based access	High	Medium	Low	Low
Bus-based access	Medium	Medium	Medium	Medium

Most of the relevant research on resource-constrained SOC testing has been focused on dedicated test bus access (due to its modularity, scalability and flexibility in trading-off different cost factors) and it was concentrated in two main directions:

- *Test scheduling and test architecture optimisation:* Given the number of TAM wires or test pins, efficient test architectures and test schedules are determined in a short development time, under constraints such as test power and physical layout. Numerous algorithms have been proposed on this topic and they are surveyed in Section 3.
- *Test resource partitioning:* By moving some test resources from ATE to the SOC, both the volume of test data that needs to be stored in the ATE buffers and the SOC testing time can be reduced. We overview the relevant work in this direction in Section 4.

3 Test scheduling and test architecture optimisation

The optimisation of modular test architectures (e.g. Test Bus and TestRail) and test scheduling have been subject to extensive research. For an SOC with specified parameters for its cores, a test architecture and a test schedule are designed to minimise the test cost, which must account for the test application time and the amount of DFT on-chip resources (both logic and routing).

Various constraints need to be considered during test scheduling, which is the process that determines the start and end test times for all the cores. For example, testing more cores in parallel usually can decrease the TAT, however it will also increase the test power, which may lead to destructive testing [37]. In addition, *concurrency constraints* may exist due to sharing of test resources (e.g. a wrapped core cannot be tested at the same time with its surrounding unwrapped UDL because both of them use the wrapper

boundary cells during test). Furthermore, in certain cases a user-defined partial ordering, called *precedence constraints*, is helpful in test scheduling. For instance, the core tests which are more likely to fail can be scheduled in front of the core tests that are less likely to fail. This helps to reduce the average test application time when an "abort-at-first-fail" strategy is used [38]. Because test architecture optimisation and test scheduling are strongly interrelated, next we formulate the integrated test architecture optimisation and test scheduling problem (P_{taos}) as follows:

Problem P_{taos}*:* Given the number of available test pins N_p for the SOC, the peak test power constraint P_{peak}, the user-defined precedence constraint U_{prec}, the test set parameters for each core, including the number of input terminals i_c, the number of output terminals o_c, the number of test patterns p_c, the number of scan chains s_c and for each scan chain k the length of it $l_{c,k}$ (for cores with fixed-length internal scan chains) or the total number of scan flip-flops f_c (for cores with flexible-length internal scan chains), determine the wrapper design for each core, the TAM resources assigned to each core and a test schedule for the entire SOC such that: (*i*) the sum of the TAM width used at any time and the test control pin count does not exceed N_p; (*ii*) power, concurrency and precedence constraints are met; and (*iii*) the SOC test cost $C_{test} = \alpha \cdot T + (1 - \alpha) \cdot A$ is minimised, in which T is the SOC testing time, A is the DFT cost of the test architecture and α is a parameter specified by the system integrator in optimisation.

Problem P_{taos} is strong NP-hard. This is because, when $\alpha = 1$ and there are no power, concurrency and precedence constraints, problem P_{taos} is reduced to problem P_{NPAW} in [39], which was proven to be NP-hard. Because of its complexity, only a part of P_{taos} was addressed in individual prior works. We start by reviewing various test scheduling techniques with fixed DFT hardware, then we continue with wrapper design methods and we conclude with the integrated wrapper/TAM co-optimisation and test scheduling approaches.

3.1 Test scheduling

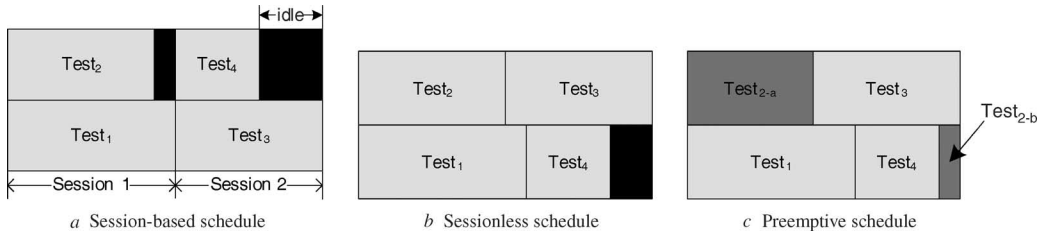
Test scheduling is the process that allocates test resources (i.e. TAM wires) to cores at different time in order to minimise the overall test application time, while at the same time satisfying the given constraints. As shown in Fig. 4, test scheduling techniques can be divided into [40, 41]:

- Session-based testing
- Sessionless testing with run to completion.
- Preemptive testing, in which a core's test can be interrupted and resumed later.

Early session-based test scheduling techniques such as [37, 42] result in long test application time. This is because the division of the schedule into test sessions leads to large idle times, as shown in Fig. 4a. Hence, most of the proposed approaches for SOC testing fall into the other two sessionless test schemes. Preemptive testing (e.g. [43, 44]) can decrease test application time. It is useful in particular in constraint-driven test scheduling where there is more idle blocks in the schedule (due to additional constraints) and an entire core test is frequently not able to fit in these idle blocks. It should be noted, however, not all core test (e.g. memory BIST) can be preempted. In addition, the system integrator must consider the added overhead caused by the use of test preemption, i.e. the extra control and time to stop and resume the preemptive core tests.

Several techniques for SOC test scheduling, independent of TAM optimisation, have been proposed in the literature. Sugihara *et al.* [45] addressed the problem of selecting a test set for each core from a set of test sets provided by the core vendors, and then schedule the test sets in order to minimise the test application time. The problem was formulated as a combinatorial optimisation problem and solved using a heuristic method. It is assumed that each core has its own BIST logic and external test can only be carried out for one core at a time. Chakrabarty [46] generalised the problem and assumed the existence of a test bus. He formulated the problem as an m-processor open shop scheduling problem and solved it using a mixed-linear integer programming (MILP) technique. Marinissen [47] addressed test scheduling problem from the test protocol expansion point of view. A test protocol is defined at the terminals of a core and describes the necessary and sufficient conditions to test the core. He modelled the test scheduling problem as a No-Wait Job Shop scheduling problem and solved it using a heuristic.

Power-constrained test scheduling is essential in order to avoid destructive testing. Zorian [37] presented a power-constrained BIST scheduling process by introducing a distributed BIST controller. In [42], Chou *et al.* proposed a method based on approximate vertex cover of a resource-constrained test compatibility graph. Muresan *et al.* [48] modelled the power-constrained scheduling problem as job scheduling problem under the constraint some jobs must be executed exclusively. Then left-edge algorithm, list scheduling and force-directed scheduling are proposed to solve the problem. Larsson and Peng [49] tried to reorganise scan chains to trade-off scan time against power consumption. Ravikumar *et al.* [50] proposed a polynomial-time algorithm for finding an optimum power-constrained schedule which minimises the testing time. Later in [51], the authors considered SOCs composed of soft and/or firm cores, and presented an algorithm for simultaneous core selection and test scheduling. In both papers, they assumed that BIST is the only methodology for testing individual cores. Zhao and Upadhyaya [52] considered the test resources (e.g. test buses and BIST engines) as queues and the core tests to be scheduled as the job entering corresponding queue. The power-constrained test scheduling problem was then formulated as the single-pair shortest path problem by representing vertices as core tests, directed edges between vertices as a segment of a schedule sequence, and the edge weight as the core testing time at the end of the segment. An efficient heuristic was proposed to solve this problem.



a Session-based schedule *b* Sessionless schedule *c* Preemptive schedule

Fig. 4 *Test scheduling technique categorisation*

3.2 Wrapper design and optimisation

To facilitate modular test development, the IEEE P1500 standard for embedded core test (SECT) [53] focuses on standardising the core test knowledge transfer and the test access to embedded cores [54]. The IEEE P1500 wrapper, as shown in Fig. 5, is a thin shell around a core that allows the core and its environment to be tested independently. The wrapper has three main modes [54, 55]: (*i*) functional operation, in which the wrapper is transparent; (*ii*) *inward-facing* test modes, in which test access is provided to the core itself; and (*iii*) *outward facing* test modes, in which test access is provided to the circuitry outside the core. The wrapper has a mandatory one-bit input/output pair, *wrapper serial input* (WSI) and *wrapper serial output* (WSO), and optionally one or more multi-bit input/output pairs, *wrapper parallel input* (WPI) and *wrapper parallel output* (WPO). The wrapper also comprises a *wrapper boundary register* (WBR) to provide controllability and observability for the core terminals and a *wrapper bypass register* (WBY) to serve as a bypass for the test data access mechanism. In addition, the wrapper has a *wrapper serial control* (WSC) port and an internal *wrapper instruction register* (WIR), used to control the different modes of the wrapper. It is important to note that IEEE P1500 SECT standardises only the wrapper interface. Hence, the internal structure of the wrapper can be adapted to the specific SOC requirements.

Core wrapper design mainly involves the construction of wrapper scan chains (called Wrapper SCs). A wrapper SC usually comprises a number of wrapper boundary cells and/or core internal scan chains. The number of wrapper SCs is equal in number to the TAM width. Since the test application time of a core is dependent on the maximum wrapper SC length, the main objective in wrapper optimisation is to build balanced wrapper SCs. For soft cores or firm cores, in which the internal scan chain design is not decided yet or can still be changed, balanced wrapper SCs are guaranteed to be constructed (lock-up latches may need to be inserted when wrapper SCs contain memory elements from multiple clock domains). For hard cores, in which the implementation of the internal scan chains is fixed, wrapper SCs are constructed by concatenating the set of core internal scan chains and wrapper boundary cells.

Marinissen *et al.* [56] first addressed the wrapper optimisation problem of designing balanced wrapper SCs for hard cores. Two polynomial-time heuristics were proposed. The *LPT* (Largest Processing Time) heuristic, originally used for the Multi-Processor Scheduling problem, was adapted to solve the wrapper optimisation problem in a very short computational time. Since the number of internal scan chains and core I/Os are typically small, the computational complexity of the LPT heuristic is quite small. The authors proposed another *COMBINE* heuristic which obtained better results by using LPT as a starting solution, followed by a linear search over the wrapper SC length with the First Fit Decrease heuristic. Iyengar *et al.* [39] proposed the *Design_wrapper* algorithm based on the Best Fit Decreasing heuristic for Bin Packing problem, which tries to minimise core test application time and required TAM width at the same time. They also showed an important feature of wrapper optimisation for hard cores, i.e. the test application time varies with TAM width as a "staircase" function. According to this feature, only a few TAM widths between 1 and W_{max}, the maximum number of TAM width, are relevant when assigning TAM resources to hard cores and these discrete widths are called *Pareto-optimal* points.

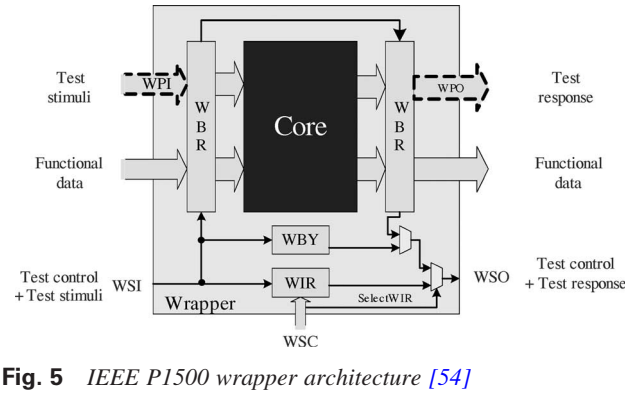


Fig. 5 *IEEE P1500 wrapper architecture [54]*

Koranne [57, 58] described a design of reconfigurable core wrapper which allows a dynamic change in the TAM width while executing the core test. This is achieved by placing extra multiplexers at the input and output of each reconfigurable scan chain. He also presented a procedure for the automatic derivation of these multiplexers using a graph representation of core wrappers. Instead of connecting the outputs of each scan chain to multiplexers, Larsson and Peng [44] presented a reconfigurable power-conscious core wrapper by connecting the inputs of each scan chain to multiplexers. This approach combines the reconfigurable wrapper [57, 58] with the scan chain clock gating [59] concepts. The reconfigurable core wrapper is useful for cores with multiple tests, where each of the tests has different TAM width requirements. For example, a core might have three tests: a BIST test for internal memory, a scan test for stuck-at faults, and a functional test for some un-modelled detects. The scan test needs a higher test bandwidth than the BIST test, while the requirement of the functional test is in the middle.

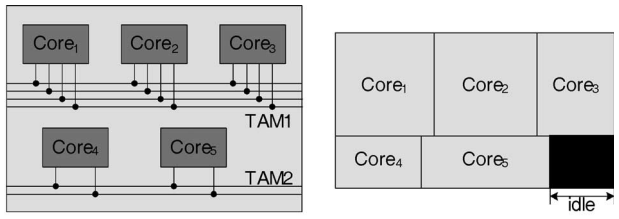
In [60], Vermaak and Kerkhoff presented a P1500-compatible wrapper design for delay fault testing, based on the digital oscillation test method. To be able to use this method, they introduced extra multiplexers and a cell address register to each wrapper cell. This approach for delay testing is only suitable for combinational cores, because only paths between the core's inputs and outputs are tested. Xu and Nicolici [61] proposed a novel core wrapper that addressed the testability problems raised by embedded cores with multiple clock domains. By partitioning core I/Os and scan cells from different clock domains into different virtual cores, they proposed to build wrapper SCs within virtual cores to solve the clock skew problem during the shift phase. To avoid clock skew during the capture phase, a capture window design technique similar to [62] was proposed that supports multi-frequency at-speed testing. The authors also described a wrapper optimisation algorithm that can tradeoff between the number of tester channels, test application time, area overhead and power dissipation. In [32], Goel described a wrapper architecture for hierarchical cores, which allows for parallel testing of the parent and child cores, at the cost of an expensive wrapper cell design.

Since wrapper design, TAM optimisation and test scheduling all have direct impacts on the SOC test cost, they should be considered in conjunction to achieve the best result. Consequently, the next Section will review the state-of-the-art techniques for the integrated problem.

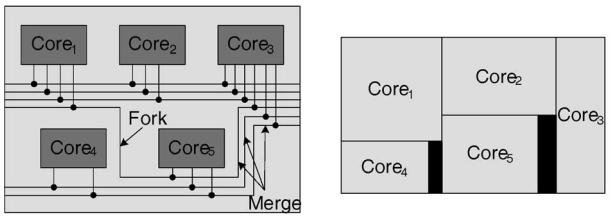
3.3 Integrated wrapper/TAM co-optimisation and test scheduling

The bus-based TAM architecture can be categorized into two types [63]:

- *Fixed-width test bus architecture*, in which the total TAM width is partitioned among several test buses with



a Fixed-width TAM bus



b Flexible-width TAM bus

Fig. 6 *TAM bus categorisation*

fixed-width, as shown in Fig. 6*a*. It operates at the granularity of TAM buses and each core in the SOC is assigned to exactly one of them.

- *Flexible-width test bus architecture*, in which TAM wires are allowed to fork and merge instead of just partitioning, as shown in Fig. 6*b*. It operates at the granularity of TAM wires and each core in the SOC can get assigned any TAM width as needed.

A number of approaches have been proposed for optimizing both architectures. Iyengar *et al.* [64] advocated the flexible-width architecture to be more effective because it improves the TAM wire utilisation. They argued that cores are frequently not assigned with a Pareto-optimal TAM width for fixed-width architecture, which wastes part of the test resources. Another reason is that test scheduling is usually more tightly integrated with TAM design in flexible-width architecture design, while in fixed-width architecture design it is often performed after TAM design by shifting tests back and forth to satisfy the given constraints. However, due to the complexity of the SOC test problem it cannot be generalised that the test cost of the flexible-width architecture is lower than the cost of the fixed-width architecture. This is because of the following reasons. Firstly, the Pareto-optimal TAM width only exists in hard cores for which the internal scan chain design is fixed. For soft or firm cores, there is no waste of test resources when assigning them to any fixed-width TAM bus. Secondly, because of the strong NP-hard attribute of P_{taos}, the size of the solution space for this problem is enormous, even with the fixed-width TAM constraint. Hence, the effectiveness of an approach is to a large extent dependent on the effectiveness of the search algorithm embedded in the approach instead of the architecture. Thirdly, and most importantly, because cores on the same TAM bus can share the same scan enable signal in fixed-width architecture, the test control pin requirement is usually lower for the fixed-width architecture than for the flexible-width architecture. These test control pins will reduce the available TAM width for test data transfer [65], which will impact significantly the SOCs with a large number of cores.

3.3.1 Fixed-width test bus architecture optimisation: The main optimisation techniques for fixed-width test bus architecture include integer linear programming (ILP), graph-based heuristics, and merge-and-distribution heuristics.

Iyengar *et al.* [39] first formulated the integrated wrapper/TAM co-optimisation problem and broke it down into a progression of four incremental problems in order of increasing complexity. For the fixed-width architecture, core assignment to a TAM bus can be represented as a binary value, and the width of each TAM partition is an integer value between 1 and W_{max}, where W_{max} is the maximum TAM width. They can be treated as variables in an ILP model. The main drawback of the ILP method is that the computation time increases exponentially because of the intractability of the problem and hence the method is not suitable for SOCs with a large number of cores and/or TAM partitions. To decrease the CPU running time, the same authors proposed to combine efficient heuristics and ILP methods [66]. By pruning the solution space the computation time is significantly reduced. However, the proposed approach returned in longer test application time than ILP in most cases. In [67], Sehgal *et al.* presented another optimisation method based on Lagrange multipliers and achieved better results.

Instead of optimising the finish time of the last core test, Koranne [68] proposed to optimise the average completion time of all the TAM partitions. Then he reduced this revised problem to the minimum weight perfect bipartite graph matching problem (not NP-hard) and proposed a polynomial time method to solve it. He also considered the TAM partitioning problem within the search space of his method, by projecting the ratio of the bitwidth [Note 1] requirement for each core to the total bitwidth of the SOC (i.e. the sum of all cores' bitwidths). However, this projection method introduces the restriction that the overall TAM width of the SOC must exceed the total number of cores. By modelling the test source/sink as source/sink in a network, TAMs as channels with capacity proportional to their width, and each core under test as a node in the network, Koranne [69] formulated the test scheduling problem as a network transportation problem. The overall test application time of the SOC is thought of as the time taken to transport the test data between cores and the test source/sink, assuming the capture time is negligible as compared to the scan shifting time. A 2-approximation algorithm was proposed to solve this problem by using the results of single source unsplittable flow problem.

While the above approaches concentrate on Test Bus architecture, Goel and Marinissen [70] addressed the same problem for fixed-width TestRail architecture with the constraint that the total TAM width must exceed the number of embedded cores inside the SOC. This constraint is removed in [71] and a new efficient heuristic *TR – Architect* is presented, which works for both cores having fixed-length and cores having flexible-length scan chains. Next, the same authors extended *TR – Architect* in [35] to support both Test Bus and TestRail architectures. They also presented the lower bounds of SOC test application time in this work. *TR – Architect* has four main steps. The basic idea is to divide the total TAM width over multiple cores based on their test data volume. The algorithm first creates an initial test architecture by assigning value 1 to each core's TAM width. Since the overall test application time of the SOC T_{soc} equals the *bottleneck* TAM with the longest test application time, in the second and the third steps, the algorithm iteratively optimises T_{soc} through merging TAMs and distributing freed TAM resources. Either two

Note 1: Bitwidth is the minimum TAM width value beyond which there is no decrease in the testing time for a given core.

non-bottleneck TAMs are merged with less TAM width to release freed TAM resources to the bottleneck TAM, or the bottleneck TAMs is merged with another TAM to decrease T_{soc}. In the last step the algorithm tries to further minimise T_{soc} by placing one of the cores assigned to the bottleneck TAM to another TAM.

In [72], Goel and Marinissen extended their *TR − Architet* algorithm to minimise both testing time and wire length. For a given TAM division, based on a proposed simple yet effective wire length cost model, they described a heuristic to determine the ordering of the cores on each TAM bus so that the overall TAM wire length is minimised. By including the wire length cost into the optimisation procedure of *TR − Architect*, a test architecture that can co-optimise the time and routing overhead was obtained. Later in [65], the same authors discussed the influence of test control on test architecture optimisation. Given the more practical test pin constraint N_p instead of total TAM width constraint W_{max}, the *TR − Architect* is extended again to a control pin-constrained version. Finally, Krundel *et al.* [73] described a Test Architecture Specification language, in which the user can express various test constraints. They then modified *TR − Architect* to incorporate the ability to satisfy these user constraints, by providing them as inputs to the tool.

3.3.2 Flexible-width test bus architecture optimisation:

For flexible-width test bus architectures, several core test representations and their corresponding optimisation techniques are summarized next.

Huang *et al.* [74] mapped the TAM architecture design to the well-known two-dimensional bin packing problem. Each core was modelled as a rectangle, in which its height corresponds to the test application time, its width corresponds to the TAM width assigned to the core, and its weight corresponds to the power consumption during test. The objective is to pack these rectangles into a bin of fixed width (SOC pins), such that the bin height (total test application time) is minimised, while not exceeding the power constraint. A heuristic method based on the Best Fit algorithm was then proposed to solve the problem. Later in [75], the authors described a new heuristic which gives better results. Another advantage of this method is that it supports multiple tests for each core. Iyengar *et al.* [63] described another heuristic for the rectangle packing problem without considering power constraints. Cores with fixed-length scan chains are assumed in this work. By exploiting the feature of Pareto-optimal TAM widths for these cores and through a series of optimisation steps, their method was shown to be more effective. Next in [76], they extended their algorithm to incorporate precedence and power constraints, while allowing a group of tests to be preemptable. They also discussed the relationship between TAM width and tester data volume in this work, so that the system integrator can identify an effective total TAM width for the SOC. Later in [77], the same authors considered minimising the ATE buffer reloads and multi-site testing, again by extending their rectangle packing algorithm. Since idle time on a TAM wire between useful bits appears as idle bits on the corresponding ATE channel, they tried to move these idle time to the end of the TAM as suggested also in [78], so that these bits do not need to be stored in the ATE. Although the test application time might be increased for a single chip, the total test data volume is reduced and hence the average testing time can be decreased when multi-site testing is employed. In [79], Zou *et al.* used *sequence pairs* to represent the placement of the rectangles, borrowed from the place-and-route literature [80]. They then employed the simulated annealing optimisation technique to find a satisfactory test schedule by altering an initial sequence pair and rectangle transformation. Their approach is shown to be more effective in terms of test application time than earlier methods, although, constraint-driven scheduling was not considered.

To include the peak power constraint into their optimisation procedure, Huang *et al.* [81] proposed to model the core as a 3-D cube, where the TAM width, peak power and core testing time represent the three dimensions. They then formulated the integrated wrapper /TAM co-optimisation and test scheduling problem under power constraints as a restricted 3-D bin-packing problem and proposed a heuristic to solve it. Following this idea, when additional constraints exist, the problem formulation can be extended as a bin-packing problem with appropriate number of dimensions.

Koranne and Iyengar [82] proposed a *p*-admissible representation of SOC test schedules based on the use of k-tuples. Compared with rectangle and 3-D cube representations, *p*-admissible representation has the benefits to be readily amenable to several optimisation techniques, such as tabu-search, simulated annealing, two-exchange, and genetic algorithms. A greedy random search algorithm was presented to find an optimal test schedule.

By utilizing the reconfigurable wrapper concepts [57, 58], Koranne modelled the core test schedule as a malleable job rather than a static job [83]. A new polynomial time algorithm was then presented using a combination of a network flow algorithm [69] and a malleable job scheduling algorithm. In [44, 84], Larsson *et al.* showed the test scheduling problem is equivalent to independent job scheduling on identical machines, when a reconfigurable wrapper and preemptive scheduling are used. Consequently, a linear time heuristic, which is able to produce a close-to lower bound solution, was proposed.

In [85], Larsson and Peng presented an integrated test framework by analysing test scheduling under power and test resource constraints along with TAM design. They formulated the test architecture optimisation problem as a set of equations and proposed a polynomial-time algorithm to solve them. Later in [86], a simulated annealing algorithm was proposed to reduce the CPU run time for large SOCs. The authors are one of the first researchers who considered almost all aspects of core-based SOC testing in an unified approach. However, the problem is over simplified by the assumption of linear dependence of testing time and power on scan chain subdivision.

Zhao and Upadhyaya [87] addressed the power-constrained test scheduling problem based on a graph-theoretic formulation. They constructed the power-constrained concurrent core tests and use the test compatibility information to settle the preferred TAM width for the tests. Dynamic test partitioning techniques are proposed to dynamically assign the TAM width and reduce the explicit idle time. In [88], Su and Wu proposed another graph-based approach to solve the same problem. First the test compatibility graph (TCG) is built based on given test resource conflicts. TCG is a complete graph if no predetermined test resource conflicts exist. The authors proved that the problem can be reduced to finding a partial graph G of the TCG that satisfies: (1) G is an interval graph, (2) each maximum clique of G satisfies the power constraint, and (3) the overall test application time is minimised. To reduce the computation time of searching the solution space, a tabu search based heuristic is used for rapid exploration. In [89], instead of optimizing test schedule, Huang *et al.* proposed to use the test scheduling information as a constraint to optimise the total number of

SOC test pins. The authors formulated this constraint-driven pin mapping problem as a chromatic number problem in graph theory and as a dependency matrix partitioning problem used in pseudo-exhaustive testing. A heuristic algorithm based on clique partitioning was proposed to solve the NP-hard problem. Larsson and Fujiwara [90] presented a test resource partitioning and optimisation technique that produces a test resource specification, which can serve as an input to the test automation tool. The routing overhead was considered in the cost model, by using a single point to represent each core, and calculating the length of a TAM wire with a Manhattan distance function. The authors proposed a technique for the iterative improvement of a test specification by using Gantt charts. When the SOC contains a large number of cores and test resources, however, the search for an optimal solution will become computationally expensive.

Xu and Nicolici [91] addressed the problem of reducing the area and performance overhead introduced by wrapper boundary cells. They proposed a test architecture for SOCs with unwrapped cores, by dividing the total TAM into CUT, producer and consumer TAM groups that are used to scan in/out the internal scan chain, the primary inputs and the primary outputs, respectively. The authors also presented a heuristic to solve the test scheduling problem, by adapting the rectangle packing algorithm in [63]. Because of the reduction in DFT area, the method introduces additional test resource conflicts, which leads to an increased testing time.

3.4 Comparison of test architecture optimisation techniques

In this Section, we compare eight recently-proposed test architecture optimisation techniques in terms of the TAT on five benchmark SOCs from the ITC'02 SOC Test Benchmarks [92]: d695 and g1023 are hypothetical SOCs built from ISCAS benchmark circuits, while p22810, p34392 and p93791 are industrial SOCs. As shown in Table 2, the eight approaches include four fixed-width test bus architecture

Table 2: Comparison of eight test architecture optimisation techniques

			Fixed-width				Flexible-width			
		LB_T	ILP/enum	Par_eval	TR-Architect	Lagrange	GRP	3-D Packing	Tabu	SA_2
SOC	W_{max}	[35]	[39]	[66]	[35]	[79]	[63]	[81]	[88]	[79]
d695	16	40951	42568	42644	44307	–	44545	42716	41905	**41899**
	24	27305	28292	30032	28576	–	31569	28639	28231	**28165**
	32	20482	21566	22268	21518	–	23306	21389	21467	**21258**
	40	16388	17901	18448	17617	–	18837	17366	17308	**17101**
	48	13659	16975	15300	14608	–	16984	15142	14643	**14310**
	56	11709	13207	12941	12462	–	14974	13208	12493	**12134**
	64	10247	12941	12941	11033	–	11984	11279	11036	**10760**
g1023	16	30161	–	–	34459	–	–	31444	32602	**31398**
	24	20112	–	–	22821	–	–	21409	22005	**21365**
	32	15088	–	–	16855	–	–	16489	17422	**16067**
	40	14794	–	–	**14794**	–	–	**14794**	**14794**	**14794**
	48	14794	–	–	**14794**	–	–	**14794**	**14794**	**14794**
	56	14794	–	–	**14794**	–	–	**14794**	**14794**	**14794**
	64	14794	–	–	**14794**	–	–	**14794**	**14794**	**14794**
p22810	16	419466	462210	468011	458068	**434922**	489192	446684	465162	438619
	24	279644	361571	313607	299718	313607	330016	300723	317761	**293019**
	32	209734	312659	246332	222471	245622	245718	223462	236796	**219923**
	40	167787	278359	232049	190995	194193	199558	184951	193696	**180004**
	48	139823	278359	232049	160221	164755	173705	167858	174491	**151886**
	56	119848	268472	153990	145417	145417	157159	145087	155730	**132812**
	64	104868	260638	153990	133405	133628	142342	128512	145417	**112515**
p34392	16	932790	998733	1033210	1010821	1021510	1053491	1016640	995739	**965252**
	24	621093	720858	882182	680411	729864	759427	681745	690425	**657561**
	32	544579	591027	663193	551778	630934	**544579**	553713	**544579**	**544579**
	40	544579	**544579**	**544579**	**544579**	**544579**	**544579**	**544579**	**544579**	**544579**
	48	544579	**544579**	**544579**	**544579**	**544579**	**544579**	**544579**	**544579**	**544579**
	56	544579	**544579**	**544579**	**544579**	**544579**	**544579**	**544579**	**544579**	**544579**
	64	544579	**544579**	**544579**	**544579**	**544579**	**544579**	**544579**	**544579**	**544579**
p93791	16	1746657	1771720	1786200	1791638	1775586	1932331	1791860	1767248	**1765797**
	24	1164442	1187990	1209420	1185434	1198110	1310841	1200157	1178776	**1178397**
	32	873334	**887751**	894342	912233	936081	988039	900798	906153	893892
	40	698670	**698583**	741965	718005	734085	794027	719880	737624	718005
	48	582227	599373	599373	601450	599373	669196	607955	608285	**597182**
	56	499053	514688	514688	528925	514688	568436	521168	539800	**510516**
	64	436673	460328	473997	455738	472388	517958	459233	485031	**451472**

optimisation methods: (1) the Test Bus Architecture optimisation method based on ILP and exhaustive enumeration in [39], (2) the heuristic Test Bus Architecture optimisation method Par_eval in [66], (3) the merge-and-distribution heuristic TR-Architect in [35], and (4) the Lagrange-based method in [67]; and four flexible-width test bus architecture optimisation methods: (5) the generalised rectangle-packing-based optimisation (GRP) in [63], (6) the 3-D packing heuristic in [81], (7) the graph-based tabu search method in [88] and (8) the simulated annealing algorithm in [79]. Goel and Marinissen [35] presented an architecture-independent lower bound on the SOC TAT, as shown in the LB_T column of Table 2. When the embedded cores come with fixed-length scan chains, the lower bound stops decreasing for large enough W_{max} (e.g. $W_{max} \geq 40$ for g1023 and $W_{max} \geq 32$ for p34392). If this happens, it means that there is at least one core that forms the bottleneck and for which increasing its TAM width does not decrease the SOC TAT any more.

From Table 2, we can observe that several optimisation techniques generate close-to lower bound results. Although [79] gives better results in most cases, there are no significant improvements over the other approaches. In addition, the reported results are provided under several simplifying assumptions: (i) TAM width constraint W_{max}, instead of the more practical constraint N_p (the number of available test pins), is specified; (ii) there are no power, concurrency and precedence constraints; (iii) off-chip ATE is the only test source/sink; (iv) all scan-tested cores are equipped with fixed-length scan chains; (v) even though some of these SOCs originally contain multiple levels of design hierarchy, they are assumed to be flattened in test architecture design. When one or more of these assumptions are removed, some approaches can still be used effectively without changes or can be easily extended, whereas others may lead to undesirable results after adaptation. For example, [35] can be easily extended to a control-aware version as shown in [65]. Therefore, since we cannot safely draw a conclusion on which approach of the above is the best, the system integrators should select an approach based on their specific test requirements.

In terms of DFT, the above approaches used dedicated test busses and the standard core test wrappers. Better results can be achieved when additional DFT hardware is introduced. For example, when reconfigurable wrapper and preemptive scheduling are used, the results given in [44] are almost the same as the lower bound value. When decoder and compactor are implemented on-chip to compress test sets, not only the test data stored in the ATE can be significantly reduced, but the SOC testing time may also be dramatically lower than the lower bound presented in [33] (since the lower bound is obtained without using the DFT logic for test data compression). Therefore, the SOC test resource partitioning is discussed in the following section.

4 SOC test resource partitioning

As discussed in Section 1, BIST generates test stimuli and compares test responses on-chip. It is an alternative approach to ATE-based external testing, although, due to the limited fault coverage of pseudo-random techniques the use of external test resources is required. This leads to the new test resource partitioning approaches [93], where on-chip embedded test resources interact with the external test equipment.

Full scan is the mainstream DFT technique employed by most of the chip designers and embedded core providers. For full-scanned embedded cores, the test cubes generated by ATPG tools feature a large number of unspecified ("don't-care" or X) bits which can be assigned arbitrary values, even after static and/or dynamic compaction. Various test data compression (TDC) techniques significantly reduce the test data volume stored in ATE by exploiting these don't care bits. TDC mainly involves two parts: test stimuli compression and test response compaction. A conceptual architecture is illustrated in Fig. 7, in which the decoder is used to decompress the compressed test vectors sent from ATE and the compactor is used to compress the test responses. It should be noted that TDC only reduces the test data volume that needs to be transferred between ATE and the chip. It does not reduce the number of test patterns applied to the chip, which is different with traditional test set compaction techniques [5] and has the advantage of being able to detect un-modelled defects.

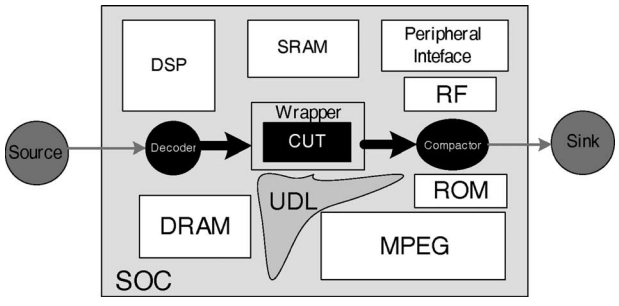


Fig. 7 *Conceptual infrastructure for SOC testing with test data compression*

4.1 Test stimuli compression

LFSR-reseeding, was first proposed by Koenemann in [6]. Instead of storing the fully specified test patterns on ATE, he proposed to store the LFSR seeds, whose size is only slightly higher than the number of the maximum care bits in the test cubes. As a follow up to this work, many approaches were proposed to improve the encoding efficiency and/or reduce the possibility of a phenomenon called "test pattern lockout" caused by the linear dependencies in the LFSR sequences. Since the encoding capability depends on the LFSR size, a large LFSR may be necessary for circuits with very large number of scan cells. In addition, when the LFSR is cycling to completely fill the scan chains (expansion in time domain), the ATE sits idle for a long period of time which leads to inefficient ATE usage. As a consequence, continuous-flow TDC techniques [94–96] were proposed and they expand the test stimuli in the spatial domain (i.e. from a low ATE channel width to a higher chip scan channel width) and hence make full use of the ATE resources. A relatively small mapping structure, e.g. XOR-based network in [94], SISR/MISR-like finite state machine and XOR-based network in [95], or ring generator and phase shifter in [96], were introduced. These decompression structures relate the inputs and outputs using a set of linear equations. Compressed test sets can be easily generated by solving the sets of equations generated using the test cube information. Instead of introducing the additional mapping logic, Dorsch *et al.* [97] proposed to make use of the scan chains from the cores in the SOC to compress the test sets for the CUT. The above techniques are embedded in the ATPG flow or exploit the core's structural information which may not be suitable for the IP-protected core-based design flow.

An effective TDC approach suitable for SOC testing is based on data compression using coding techniques, which exploit the following features.

- The "don't care" bits in test cubes can be filled with arbitrary values to form a long run of 0's or 1's.
- There is a lot of similarity between test vectors because of the structural relationship among faults in the circuit. By carefully ordering the test sets, successive test vectors will differ in only a small number of bits. Then, the information about how the vectors differ, instead of the original vectors, can be encoded to reduce test data volume.

These techniques compress test data without requiring the structural information from the core provider (which is necessary for integrated compression, ATPG and fault simulation) and therefore they fit well in the core-based design flow.

To achieve compressed SOC test, data is first decomposed into either fixed-length or variable-length blocks and a code word, also of either fixed or variable length, is assigned to each block. The basic idea is to assign frequent blocks to a comparably smaller code word. Iyengar *et al.* [98] proposed a BIST approach for testing non-scanned circuits based on statistical coding. Jas *et al.* [99] presented a compression technique for scanned circuits by dividing the test vectors into fixed-length blocks and using the Huffman coding technique. In [100], Jas and Touba described another TDC scheme using run-length coding. By exploiting the capabilities of present ATEs to assign groups of inputs to ports and to perform vector repeat per port, Vranken *et al.* [101] implemented a similar run-length coding scheme, although the decompression is achieved off-chip on the ATE instead of on-chip.

The above coding techniques are based on "variable-to-fixed-length" codes. Chandra and Chakrabarty [102] first proposed a "variable-to-variable-length" test data coding scheme, based on Golomb coding [103]. When using Golomb-coding the savings in scan-in power are trade-off against improvement in compression ratio. Rosinger *et al.* described a minimum transition count (MTC) coding scheme, which can simultaneously reduce test data and test power. Frequency-directed run-length (FDR) coding [104], another variable-to-variable-length coding technique proposed by Chandra and Chakrabarty, further increased the compression ratio. It is designed based on the observation that the frequency of runs decreases with the increase of their length. While the original FDR coding was based on encoding runs of 0's, El-Maleh and Al-Abaji [105] extended it with EFDR coding, by encoding both runs of 0's and 1's. Gonciari *et al.* [106] analysed the three main test data compression environment parameters: compression ratio, decoder area overhead and test application time, and introduced a variable-length input Huffman (VIHC) coding scheme to efficiently trade them off. Finally, Tehranipour *et al.* [107] presented a 9C coding scheme that supports mapping "don't-care" bits to random values instead of dedicated long runs of 0's and 1's, which is able to detect more non-modelled defects.

Several dictionary-based TDC techniques have also been presented recently. While the above statistical coding schemes use a statistical model of the test data and encode blocks according to their frequencies of occurrence, dictionary-based methods select strings of the blocks to establish a dictionary, and then encode them into equal-size tokens [103]. The dictionary may be either static or dynamic (adaptive). The static dictionary is permanent, whereas the dynamic dictionary permits additions and deletions of strings as new input is processed. Wolff and Papachristou [108] described a method that employs the well-known LZ77 algorithm, which uses a dynamic dictionary. Knieser *et al.* [109] presented another TDC technique based on LZW algorithm. Li and Chakrabarty [110] proposed a TDC approach using dictionaries with fixed-length indices. One advantage of this method is the elimination of the synchronisation problem between the ATE and the SOC, because it does not require multiple clock cycles to determine the end of a compressed data packet.

Unlike the above TDC techniques that require an on-chip hardware decoders, Jas and Touba [111] proposed a software decompression approach for SOCs with embedded processors. The basic idea is to load a program along with the compressed test sets into the processor's on-chip memory. The processor then executes the program to decompress the test data and applies it to the CUT. The authors also described techniques to avoid the potential problem of "memory overflow", at the cost of lower compression ratio or longer test application time. Although it saves DFT area, the embedded processors may have access only to a small number of cores and hence the approach lacks generality.

4.2 Test response compaction

Test response compaction is the process that reduces the volume of test responses sent from the CUT to the ATE. Unlike the lossless compression techniques used for test stimuli, test response compaction techniques are typically lossy, i.e. some faults that can be detected by the original test response may be masked by the compactor. This phenomenon is referred to as *aliasing*. An additional problem lies in the fact that many designs produce unknown states (also called X states) in test responses. Sources of these unknown states include bus contention, interactions between multiple clock domains or uninitialised memory elements. Although inserting test points helps to eliminate unknown states in test responses, this intrusive DFT technique is not preferable for core-based design. Further, the test response compactor should have also support for reliable diagnosis, in addition to very low aliasing and the ability to handle responses with unknown states.

Based on their structures, test response compactors can be categorized into three types [112]: infinite memory compactors (also called time compactors or sequential compactors), memoryless compactors (also called space compactors or combinational compactors) and finite memory compactors.

Infinite memory compactors, such as MISRs or cellular automata, have been widely used in BIST environments [5]. By inserting an MISR between the scan chain outputs and bi-directional scan pins, Barnhart *et al.* [113] presented a method to compact scan data during the scan-out operation. A characteristic feature of the above compactor is the infinite impulse response property, which allows the compaction ratios to be a huge number from 10^6 to 10^8 with a very small aliasing possibility. If the test responses contain unknown states, these unknown states will need to be masked to avoid the corruption of the response signature.

Memoryless compactors are combinational circuits. They compact a large number of scan outputs, m, to a much smaller number, k, through a compaction circuitry (usually XOR networks). Various linear and nonlinear compactors have been proposed in the literature. Mitra and Kim [114] proposed X-Compact technique that uses parity check matrices where all the columns have the same weight (the weight of a column is the number of 1 s in it). Their approach can reach very high compaction ratio (for memoryless compactors), and at the same time handle simultaneous errors and unknown values. I-compact [115] is based on coding theory of error detection in the presence of

erasures. The approach uses parity check matrices of a linear block code (e.g. Hamming codes) to achieve compaction of test response data with many unknowns. Memoryless compactors can handle unknown states in test responses without circuit modification. However, the compaction ratio is much smaller when compared with the infinite memory compactors.

Finite memory compactors contain memory elements, however they have a finite impulse response. In [112], Rajski *et al.* first introduced this type of compactor: the convolutional compactor. The finite memory compactors can achieve compression ratios in between that of memoryless and infinite memory compactors. They also can handle unknown states and support diagnosis, at the cost of a slightly higher DFT area.

5 Ongoing and future research

Although various techniques have been proposed to tackle the resource-constrained SOC test problem, with the growing complexity of SOC designs and the aggressive shrinking of semiconductor technologies, new methodologies are still required to be developed to address the emerging challenges.

Most of the prior research on TAM optimisation and test scheduling assumes that the ATE operates at the same frequency (f_{ate}) as the internal cores' scan chain frequencies (f_{tam}). This assumption has been shown to be inefficient when there is a mismatch between the ATE capability and the internal scan chain speed. In [116] Khoche proposed the *Bandwidth Matching* technique to tackle the mismatch between high-end ATEs and low-speed scan chain designs. Bandwidth is defined as the product of the width and the frequency of a scan architecture. Using serialisation/deserialisation methods, a high bandwidth source/sink (e.g. ATE) can be connected to multiple low bandwidth sinks/sources (e.g. TAM), as long as the bandwidth matches. There are also cases where low-speed ATEs are used to feed high-speed scan chains, as emphasized in [117]. The TAM frequency has direct impact on test application time, TAM wire length and test power of the SOC. Therefore, the system integrator will have to search a larger solution space to explore the trade-off between these important factors by including the TAM frequency into the test architecture optimisation. Sehgal *et al.* have proposed virtual TAM design using a framework based on Lagrange multipliers [67] to fully exploit the high bandwidth capacity of the ATE. The approach is based on a given frequency ratio between the ATE channels and the internal scan frequency. However, the frequency was not considered as an integrated optimisation component. Therefore, multi-frequency TAM design, in which both the TAM width and the TAM running frequency assigned to a core are co-optimised, is necessary to further reduce the SOC test cost.

Both test architecture optimisation and TDC techniques affect the test application time of the SOC. Therefore, to further reduce the cost of SOC testing, these two tasks should be considered simultaneously. For example, a higher compression ratio of a core test will require less TAM bandwidth. If test architecture optimisation is applied to the uncompressed test sets, the core will be given a higher TAM width than needed and hence an inefficient test architecture will be derived. In [118], Iyengar and Chandra considered both techniques by first designing the TAM architecture using the rectangle packing algorithm [63] for the original test sets, and then applying the FDR coding [104] to cores for test compression purpose. Although both techniques are applied, they are not considered in a unified manner and results show that large idle times exist in the test schedule. There can be three methods to combine decompression logic (i.e. decoder) with TAM design: decoder-per-SOC, decoder-per-TAM and decoder-per-core, in which decoder-per-TAM is only applicable for fixed-width test bus architectures. The system integrator needs to trade-off the DFT area and test application time within these three scenarios. Since different core tests typically have different test compression ratios, decoder-per-core strategy will achieve the best overall test data reduction, although, at the cost of the largest DFT overhead. Decoder-per-SOC, on the contrary, will have the least DFT area overhead but the larger test data volume. Decoder-per-TAM is in the middle of the above two strategies. Despite the intuitive analysis, how to efficiently combine the test architecture optimisation with any of the three scenarios is still an open issue and needs further investigation.

More defects in deep sub-micron technologies manifest themselves through unexpected increases or decreases in propagation delays, known as delay faults [119]. To detect these timing-related defects, at least two controlled patterns in consecutive clock cycles are necessary: the first initialises the circuit's nodes and the second captures the transitions on the sensitized paths. When applying these two consecutive patterns, the faulty propagation paths will not be detected unless the outputs are sampled at a high enough speed (preferably rated-speed). Most of the research work on bus-based test architectures is applicable only to one-pattern tests, which is not suitable for delay tests. In [120], Xu and Nicolici proposed to reuse the functional interconnects for applying the second pattern to the primary inputs of the CUT, with a penalty in testing time. This approach, however, need to be extended for at-speed application of the second test pattern. Therefore, new techniques are needed to address the increasing demand for high-quality delay test. In addition, future challenges include hierarchical TAM design, test architecture optimisation with multiple test sources and/or test sinks, and test architecture optimisation for network-on-a-chip designs.

6 References

1 International SEMATECH, 'The international technology roadmap for semiconductors (ITRS)', 2001 Edition, http://public.itrs.net/Files/2001ITRS/Home.htm
2 Gupta, R.K., and Zorian, Y.: 'Introducing core-based system design', *IEEE Des. Test Comput.*, 1997, **14**, (4), pp. 15–25
3 Zorian, Y., Marinissen, E.J., and Dey, S.: 'Testing embedded-core-based system chips', *Computer*, 1999, **32**, (6), pp. 52–60
4 Kapur, R., and Williams, T.W.: 'Manufacturing test of SoCs'. ATS, 2002, pp. 317–319
5 Abramovici, M., Breuer, M., and Friedman, A.: 'Digital systems testing and testable design' (IEEE Press, 1990)
6 Koenemann, B.: 'LFSR-coded test patterns for scan designs'. ETC, 1991, pp. 237–242
7 Iyengar, V., Chakrabarty, K., and Marinissen, E.J.: 'Recent advances in test planning for modular testing of core-based SOCs'. ATS, 2002, pp. 320–325
8 Rajsuman, R.: 'Testing a system-on-a-chip with embedded microprocessor'. ITC 1999, pp. 499–508
9 Zorian, Y.: 'Test requirements for embedded core-based systems and IEEE P1500'. ITC 1997, pp. 191–199
10 Zorian, Y., Marinissen, E.J., and Dey, S.: 'Testing embedded-core based system chips'. ITC 1998, pp. 130–143
11 Immaneni, V., and Raman, S.: 'Direct access test scheme - design of block and core cells for embedded ASICs'. ITC 1990, pp. 488–492
12 Bhatia, S., Gheewala, T., and Varma, P.: 'A unifying methodology for intellectual property and custom logic testing'. ITC, 1996, pp. 639–648
13 Whetsel, L.: 'An IEEE 1149.1 Based test access architecture for ICs with embedded cores'. ITC 1997, pp. 69–78
14 Touba, N., and Pouya, B.: 'Using partial isolation rings to test core-based designs', *IEEE Des. Test Comput.*, 1997, **14**, (4), pp. 52–59
15 Pouya, B., and Touba, N.: 'Modifying user-defined logic for test access to embedded cores'. ITC, 1997, pp. 60–68

16 Harrod, P.: 'Testing reusable IP - a case study'. ITC, 1999, pp. 493–498
17 Beenker, F., van Eerdewijk, K., Gerritsen, R., Peacock, F., and van der Star, M.: 'Macro testing: unifying IC and board test', *IEEE Des. Test Comput.*, 1986, **3**, (4), pp. 26–32
18 Bouwman, F., Oostdijk, S., Stans, R., Bennetts, B., and Beenker, F.: 'Macro testability; the results of production device applications'. ITC, 1992, pp. 232–241
19 Marinissen, E.J., and Lousberg, M.: 'Macro test: A liberal test approach for embedded reusable cores'. TECS, 1997, pp. 1.2–1–9
20 Ghosh, I., Jha, N.K., and Dey, S.: 'Low overhead design for testability and test generation technique for core-based systems-on-a-chip', *IEEE Trans. Comput.-Aided Des.*, 1999, **18**, (11), p. 1661
21 Freeman, S.: 'Test generation for data-path logic: the F-path method', *IEEE J. Solid-State Circuits*, 1988, **23**, pp. 421–427
22 Ghosh, I., Dey, S., and Jha, N.K.: 'A fast and low cost testing technique for core-based system-chips', *IEEE Trans. Comput.-Aided Des.*, 2000, **19**, (8), pp. 863–876
23 Ravi, S., Lakshminarayana, G., and Jha, N.K.: 'Testing of core-based systems-on-a-chip', *IEEE Trans. Comput.-Aided Des.*, 2001, **20**, (3), pp. 426–439
24 Ravi, S., and Jha, N.K.: 'Test synthesis of system-on-a-chip', *IEEE Trans. Comput.-Aided Des.*, 2002, **21**, (10), pp. 1211–1217
25 Dick, R.P., and Jha, N.K.: 'MOCSYN: multiobjective core-based single-chip system synthesis'. DATE, 1999, 263–270
26 Yoneda, T., and Fujiwara, H.: 'Design for consecutive testability of system-on-a-chip with built-In self testable cores', *J. Electron. Test., Theory Appl.*, 2002, **18**, (4/5), pp. 487–501
27 Chakrabarty, K., Mukherjee, R., and Exnicios, A.S.: 'Synthesis of transparent circuits for hierarchical and system-on-a-chip test'. ICVD, 2001, pp. 431–436
28 Yoneda, T, and Fujiwara, H.: 'Design for consecutive transparency of cores in system-on-a-chip'. VTS, 2003, pp. 287–292
29 Yoneda, T, Uchiyama, T., and Fujiwara, H.: 'Area and time co-optimization for system-on-a-chip based on consecutive testability'. ITC, 2003, Charlotte, NC, Sept. 2003, pp. 415–422
30 Nourani, M., and Papachristou, C.: 'Structural fault testing of embedded cores using pipelining', *J. Electron. Test., Theory Appl.*, 1999, **15**, (1), p. 129
31 Aerts, J., and Marinissen, E.J.: 'Scan chain design for test time reduction in core-based ICs'. ITC, 1998, pp. 448–457
32 Goel, S.K.: 'An improved wrapper architecture for parallel testing of hierarchical cores'. ETS, 2004, pp. 147–152
33 Varma, P., and Bhatia, S.: 'A structured test re-use methodology for core-based system chips'. ITC, 1998, pp. 294–302
34 Marinissen, E.J., Arendsen, R., Bos, G., Dingemanse, H., Lousberg, M., and Wouters, C.: 'A structured and scalable mechanism for test access to embedded reusable cores'. ITC, 1998, pp. 284–293
35 Goel, S.K., and Marinissen, E.J.: 'Effective and efficient test architecture design for SOCs'. ITC, 2002, pp. 529–538
36 Nag, P., Gattiker, A., Wei, S., Blanton, R., and Maly, W.: 'Modeling the economics of testing: a DFT perspective', *IEEE Des. Test Comput.*, 2002, **19**, pp. 29–41
37 Zorian, Y.: 'A distributed BIST control scheme for complex VLSI devices'. VTS, 1993, pp. 6–11
38 Jiang, W., and Vinnakota, B.: 'Defect-oriented test scheduling'. VTS, 1999, pp. 433–438
39 Iyengar, V., Chakrabarty, K., and Marinissen, E.J.: 'Co-optimization of test wrapper and test access architecture for embedded cores', *J. Electron. Test., Theory Appl.*, 2002, **18**, (2), pp. 213–230
40 Flottes, M.-L., Pouget, J., and Rouzeyre, B.: 'Sessionless test scheme: power-constrained test scheduling for system-on-a-chip'. VLSI-SOC 2001, pp. 105–110
41 Larsson, E., and Fujiwara, H.: 'Power constrained preemptive TAM scheduling'. ETW, 2002, pp. 119–126
42 Chou, R.M., Saluja, K.K., and Agrawal, V.D.: 'Scheduling tests for VLSI systems under power constraints', *IEEE Trans. VLSI Syst.*, 1997, **5**, (2), pp. 175–184
43 Iyengar, V., and Chakrabarty, K.: 'Precedence-based, preemptive, and power-constrained test scheduling for system-on-a-chip'. VTS, 2001, pp. 368–374
44 Larsson, E., and Peng, Z.: 'A reconfigurable power-conscious core wrapper and its application to SOC test scheduling'. ITC, 2003, pp. 1135–1144
45 Sugihara, M., Date, H., and Yasuura, H.: 'A novel test methodology for core-based system LSIs and a ddddtesting time minimization problem'. ITC, 1998, pp. 465–472
46 Chakrabarty, K.: 'Test scheduling for core-based systems using mixed-integer linear programming', *IEEE Trans. Comput.-Aided Des.*, 2000, **19**, (10), pp. 1163–1174
47 Marinissen, E.J.: 'The role of test protocols in automated test generation for embedded-core-based system ICs', *J. Electron. Test., Theory Appl.*, 2002, **18**, (4/5), pp. 435–454
48 Muresan, V., Wang, X., Muresan, V., and Vladutiu, M.: 'A comparison of classical scheduling approaches in power-constrained block-test scheduling'. ITC, 2000, pp. 882–891
49 Larsson, E., and Peng, Z.: 'Test scheduling and scan-chain division under power constraint'. ATS, 2001, pp. 259–264
50 Ravikumar, C.P., Verma, A., and Chandra, G.: 'A polynomial-time algorithm for power constrained testing of core based systems'. ATS, 1999, 1999, pp. 107–112
51 Ravikumar, C.P., Chandra, G., and Verma, A.: 'Simultaneous module selection and scheduling for power-constrained testing of core based systems'. VLSID, 2000, pp. 462–467
52 Zhao, D., and Upadhyaya, S.: 'A generic resource distribution and test scheduling scheme for embedded core-based SoCs', *IEEE Trans. Instrum. Meas.*, 2004, **53**, (2), pp. 318–329
53 Hales, A., and Marinissen, E.J.: IEEE P1500 Web Site, http://grouper.ieee.org/groups/1500/
54 Marinissen, E.J., Kapur, R., Lousberg, M., Mclaurin, T., Ricchetti, M., and Zorian, Y.: 'On IEEE P1500's standard for embedded core test', *J. Electron. Test., Theory Appl.*, 2002, **18**, (4/5), pp. 365–383
55 Marinissen, E.J., Kapur, R., and Zorian, Y.: 'On using IEEE P1500 SECT for test plug-n-play'. ITC, 2000, pp. 770–777
56 Marinissen, E.J., Goel, S.K., and Lousberg, M.: 'Wrapper design for embedded core test'. ITC, 2000, pp. 911–920
57 Koranne, S.: 'A novel reconfigurable wrapper for testing of embedded core-based SOCs and its associated scheduling algorithm', *J. Electron Test., Theory Appl.*, 2002, **18**, (4/5), pp. 415–434
58 Koranne, S.: 'Design of reconfigurable access wrappers for embedded core based SoC test', *IEEE Trans. VLSI Syst.*, 2003, **11**, (5), pp. 955–960
59 Nicolici, N., and Al-Hashimi, B.M.: 'Multiple scan chains for power minimization during test application in sequential circuits', *IEEE Trans. Comput.*, 2002, **51**, (6), pp. 721–734
60 Vermaak, H., and Kerkhoff, H.G.: 'Enhanced P1500 compliant wrapper suitable for delay fault testing of embedded cores'. ETW, 2003, pp. 257–262
61 Xu, Q., and Nicolici, N.: 'Wrapper design for testing IP cores with multiple clock domains'. DATE, 2004, pp. 416–421
62 Hetherington, G., Fryars, T., Tamarapalli, N., Kassab, M., Hassan, A., and Rajski, J.: 'Logic BIST for large industrial designs: real issues and case studies'. ITC, 1999, pp. 358-367
63 Iyengar, V., Chakrabarty, K., and Marinissen, E.J.: 'On Using rectangle packing for SOC wrapper/TAM co-Optimization'. VTS, 2002, pp. 253–258
64 Iyengar, V., Chakrabarty, K., and Marinissen, E.J.: 'Test access mechanism optimization, test scheduling, and tester data volume reduction for system-on-chip', *IEEE Trans. Comput.*, 2003, **52**, (12), pp. 1619–1632
65 Goel, S.K., and Marinissen, E.J.: 'Controla-aware test architecture design for modular SOC testing'. ETW, 2003, pp. 57–62
66 Iyengar, V., Chakrabarty, K., and Marinissen, E.J.: 'Efficient wrapper/TAM co-optimization for large SOCs'. DATE, 2002, pp. 491–498
67 Sehgal, A., Iyengar, V., Krasniewski, M.D., and Chakrabarty, K.: 'Test cost reduction for SOCs using virtual TAMs and lagrange multipliers'. DAC, 2003, pp. 738–743
68 Koranne, S.: 'On test scheduling for core-based SOCs'. VLSID, 2002, pp. 505–510
69 Koranne, S.: 'Formulating SoC test scheduling as a network transportation problem', *IEEE Trans. Comput.-Aided Des.*, 2002, **21**, (12), pp. 1517–1525
70 Goel, S.K., and Marinissen, E.J.: 'Cluster-based test architecture design for system-on-chip'. VTS, 2002. pp. 259–264
71 Goel, S.K., and Marinissen, E.J.: 'A novel test time reduction algorithm for test architecture design for core-based system chips'. ETW, 2002, pp. 7–12
72 Goel, S.K., and Marinissen, E.J.: 'Layout-driven SOC test architecture design for test time and wire length minimization'. DATE, 2003, pp. 738–743
73 Krundel, L., Goel, S.K., Marinissen, E.J., Flottes, M.-L., and Rouzeyre, B.: 'User-constrained test architecture design for modular SOC testing'. ETS, 2004, May 2004, pp. 80–85
74 Huang, Y., Cheng, W.-T., Tsai, C.-C., Mukherjee, N., Samman, O., Zaidan, Y., and Reddy, S.M.: 'Resource allocation and test scheduling for concurrent test of core-based SOC design'. ATS, 2001, pp. 265–270
75 Huang, Y., Cheng, W.-T., Tsai, C.-C., Mukherjee, N., and Reddy, S.M.: 'Static pin mapping and SOC test scheduling for cores with multiple test sets'. ISQED, 2003, pp. 99–104
76 Iyengar, V., Chakrabarty, K., and Marinissen, E.J.: 'Integrated wrapper/TAM co-optimization, constraint-driven test scheduling, and tester data volume reduction for SOCs'. DAC, 2002, pp. 685–690
77 Iyengar, V, Goel, S.K., Chakrabarty, K., and Marinissen, E.J.: 'Test resource optimization for multi-site testing of SOCs under ATE memory depth constraints'. ITC, 2002, pp. 1159–1168
78 Gonciari, P.T., Al-Hashimi, B.M., and Nicolici, N.: 'Useless memory allocation in system-on-a-chip test: problems and solutions'. VTS, 2002, pp. 423–429
79 Zou, W., Reddy, S.M., Pomeranz, I., and Huang, Y.: 'SOC test scheduling using simulated annealing'. VTS, 2003, pp. 325–330
80 Murata, H., Fujiyoshi, K., Nakatake, S., and Kajitani, Y.: 'VLSI module placement based on rectangle-packing by the sequence-pair', *IEEE Trans. Comput.-Aided Des.*, 1996, **15**, (12), pp. 1518–1524
81 Huang, Y., Reddy, S.M., Cheng, W.-T., Reuter, P., Mukherjee, N., Tsai, C.-C., Samman, O., and Zaidan, Y.: 'Optimal core wrapper width selection and SOC test scheduling based on 3-D bin packing algorithm'. ITC, 2002, pp. 74–82
82 Koranne, S., and Iyengar, V.: 'On the Use of k-tuples for SoC Test schedule representation'. ITC, 2002, pp. 539–548
83 Koranne, S.: 'Solving the SoC test scheduling problem using network flow and reconfigurable wrappers'. ISQED, 2003, pp. 93–98
84 Larsson, E., and Fujiwara, H.: 'Optimal system-on-chip test scheduling'. ATS, 2003, pp. 306–311
85 Larsson, E., and Peng, Z.: 'An Integrated system-on-chip test framework'. DATE, 2001, pp. 138–144

86 Larsson, E., Peng, Z., and Carlsson, G.: 'The design and optimization of SOC test solutions'. ICCAD, 2001, pp. 523–530
87 Zhao, D., and Upadhyaya, S.: 'Power constrained test scheduling with dynamically varied TAM'. VTS, 2003, pp. 273–278
88 Su, C.P., and Wu, C.-W.: 'A graph-based approach to power-constrained SOC test scheduling', *J. Electron. Test., Theory Appl.*, 2004, **20**, (1), pp. 45–60
89 Huang, Y., Mukherjee, N., Tsai, C.-C., Samman, O., Zaidan, Y., Zhang, Y., Cheng, W.-T., and Reddy, S.M.: 'Constraint-driven pin mapping for concurrent SOC testing'. ASP-DAC, 2002
90 Larsson, E., and Fujiwara, H.: 'Test resource partitioning and optimization for SOC designs'. VTS, 2003, pp. 319–324
91 Xu, Q., and Nicolici, N.: 'On reducing wrapper boundary register cells in modular SOC testing', ITC, 2003, pp. 622–631
92 Marinissen, E.J., Iyengar, V., and Chakrabarty, K.: 'ITC'02 SOC test benchmarks', http://www.extra.research.philips.com/itc02socbenchm/
93 Chandra, A., and Chakrabarty, K.: 'Test resource partitioning for SOCs', *IEEE Des. Test Comput.*, 2001, pp. 80–91
94 Bayraktaroglu, I., and Orailoglu, A.: 'Test volume and application time reduction through scan chain concealment'. DAC, 2001, pp. 151–155
95 Koenemann, B., Barnhart, C., Keller, B., Farnsworth, O., and Wheater, D.: 'A smartBIST variant with guaranteed encoding'. ATS, 2001, pp. 325-330
96 Rajski, J., Kassab, M., Mukherjee, N., Tamarapalli, N., Tyszer, J., and Qian, J.: 'Embedded deterministic test for low-cost manufacturing', *IEEE Des. Test Comput.*, 2003, pp. 58–66
97 Dorsch, R., and Wunderlich, H.: 'Tailoring ATPG for embedded testing'. ITC, 2001, pp. 530–537
98 Iyengar, V., Chakrabarty, K., and Murray, B.T.: 'Built-in self testing of sequential circuits using precomputed test sets'. VTS, 1998, pp. 418–423
99 Jas, A., Ghosh-Dastidar, J., and Touba, N.A.: 'Scan vector compression/decompression using statistical coding'. VTS, 1999, pp. 114–120
100 Jas, A., and Touba, N.: 'Test vector decompression via cyclical scan chains and its application to testing core-based designs'. ITC, 1998, pp. 458–464
101 Vranken, H., Hapke, H., Rogge, S., Chindamo, D., and Volkerink, E.: 'Atpg padding and ATE vector repeat per port for reducing test data volume'. ITC, 2003, pp. 1069-1078
102 Chandra, A., and Chakrabarty, K.: 'System-on-a-chip test data compression and decompression architectures based on golomb codes', *IEEE Trans. Comput.-Aided Des.*, 2001, **30**, (3), pp. 355–368
103 Salomon, D.: 'Data compression: The complete reference' (Springer-Verlag, New York, 2000)
104 Chandra, A., and Chakrabarty, K.: 'Frequency-directed run-length (FDR) codes with application to system-on-a-chip test data compression'. VTS, 2001, pp. 114–121
105 El-Maleh, A.H., and Al-Abaji, R.H.: 'Extended frequency-directed run-length code with improved application to system-on-a-chip test data compression'. ICECS, 2001, pp. 530–537
106 Gonciari, P.T., Al-Hashimi, B.M., and Nicolici, N.: 'Improving compression ratio, area overhead, and test application time for system-on-a-chip test data compression/decompression'. DATE, 2002, pp. 604–611
107 Tehranipour, M., Nourani, M., and Chakrabarty, K.: 'Nine-coded compression technique with application to reduced pin-count testing and flexible on-chip decompression'. DATE, 2004, pp. 1284–1289
108 Wolff, F.G., and Papachristou, C.: 'Multiscan-based test compression and hardware decompression using LZ77', ITC, 2002, pp. 331–339
109 Knieser, M.J., Wolff, F.G., Papachristou, C.A., Weyer, D.J., and McIntyre, D.R.: 'A technique for high ratio LZW compression'. DATE, 2003, pp. 116–121
110 Li, L., and Chakrabarty, K.: 'Test data compression using dictionaries with fixed-length indices'. VTS, 2003, pp. 219–224
111 Jas, A., and Touba, N.: 'Using an embedded processor for efficient deterministic testing of systems-on-a-chip'. ICCD, 1999
112 Rajski, J., Tyszer, J., Wang, C., and Reddy, S.M.: 'Convolutional compaction of test responses'. ITC, 2003, pp. 745–754
113 Barnhart, C., Brunkhorst, V., Distler, F., Farnsworth, O., Keller, B., and Koenemann, B.: 'OPMISR: the foundation for compressed ATPG vectors'. ITC, 2001, pp. 748–757
114 Mitra, S., and Kim, K.S.: 'X-compact: An efficient response compaction technique for test cost reduction'. ITC, 2002, pp. 311–320
115 Patel, J.H., Lumetta, S.S., and Reddy, S.M.: 'Application of saluja-karpovsky compactors to test responses with many unknowns'. VTS, 2003, pp. 107–112
116 Khoche, A.: 'Test resource partitioning for scan architectures using bandwidth matching'. Digest of Int. Workshop Test Resource Partition, 2002, pp. 1.4.1-1.4.8
117 Heidel, D., Dhong, S. Hofstee, P., Immediato, M., Nowka, K., Silberman, J., and Stawiasz, K.: 'High speed serializing/de-serializing design-for-test method for evaluating a 1 GHz microprocessor'. VTS, 1998, pp. 234–238
118 Iyengar, V., and Chandra, A.: 'A unified SOC test approach based on test data compression and TAM design'. DFT, 2003, pp. 511–518
119 Krstic, A., and Cheng, K.-T.: 'Delay fault testing for VLSI circuits' (Kluwer Academic Publishers, 1998)
120 Xu, Q., and Nicolici, N.: 'Delay fault testing of core-based systems-on-a-chip'. DATE, 2003, pp. 744–749