Testing embedded-core-based system chips

ARTICLE *in* COMPUTER · JULY 1999

Impact Factor: 1.44 · DOI: 10.1109/2.769444 · Source: IEEE Xplore

3 AUTHORS:

Y. Zorian
Synopsys
267 PUBLICATIONS **5,096** CITATIONS
SEE PROFILE

Erik Jan Marinissen
imec Belgium
162 PUBLICATIONS **4,709** CITATIONS
SEE PROFILE

Sujit Dey
University of California, San Diego
248 PUBLICATIONS **5,888** CITATIONS
SEE PROFILE

Testing Embedded-Core Based System Chips

Yervant Zorian

LogicVision
101 Metro Drive
Third floor
San Jose, CA 95110
U.S.A.
zorian@lvision.com

Erik Jan Marinissen

Philips Research
Dept. VLSI Design Automation & Test
Prof. Holstlaan 4 - WAY-41
5656 AA Eindhoven
The Netherlands
marinis@natlab.research.philips.com

Sujit Dey

University of California, San Diego
Dept. of Electrical and Computer Eng.
9500 Gilman Drive
La Jolla, CA 92093-0407
U.S.A.
dey@ece.ucsd.edu

Abstract

Advances in semiconductor process and design technology enable the design of complex system chips. Traditional IC design, in which every circuit is designed from scratch and reuse is limited to standard-cell libraries, is more and more replaced by a design style based on embedding large reusable modules, the so-called cores. This core-based design poses a series of new challenges, especially in the domains of manufacturing test and design validation and debug. This paper provides an overview of current industrial practices as well as academic research in these areas. We also discuss industry-wide efforts by VSIA and IEEE P1500 and describe the challenges for future research.

1 Introduction

In recent years, reusable embedded modules have captured the imagination of designers who understand the potential of using such modules in building on-chip systems similar to using integrated circuits on a printed circuit board (PCB). Designers formed a rich library of pre-designed, pre-verified building blocks, the so-called *embedded cores* to import technology to a new system and differentiate the corresponding product by leveraging intellectual property advantages. Most importantly, the use of embedded cores shortened the time-to-market for new systems due to design reuse [1].

Embedded cores incorporated into system chips cover a very wide range of functions, while utilizing an unprecedented range of technologies, from CMOS logic to DRAM to analog. Cores sometimes come in hierarchical compositions; so-called *complex* cores embed one or more other, *simple*, cores. Cores come in a wide range of hardware description levels, categorized as *soft* (register-transfer level), *firm* (netlist), and *hard* (technology-dependent layout). These three types offer trade-off opportunities. Soft cores leave much of the implementation to the designer, but are flexible and process-independent. Hard cores have been optimized for predictable performance, but lack flexibility. Firm cores offer a compromise between the two. Each type of core has different modeling and test requirements [1].

However, the practical implementation of the core-based design scenario is fraught with unresolved issues: design methods for building single-chip systems, sign-off for these systems, and intellectual property licensing, protection, and liability. The most critical challenges of this emerging discipline include manufacturing test and debug. This paper analyzes these challenges and discusses the current solutions to create testable and diagnosable embedded core-based system chips. The paper covers the common practices in the industry and sheds light on the ongoing efforts to contain today's challenges. It also addresses future test challenges and research directions for system chip test.

In Section 2, the paper discusses the embedded core-based system chip test challenges. Section 3 presents a conceptual architecture for testing such system chips, consisting of three structural elements. These elements and their existing implementations are discussed in Sections 4, 5, and 6. Section 7 discusses silicon debug and diagnosis for core-based system chips. Section 8 covers the ongoing industry-wide cooperation and standardization efforts in this domain. Section 9 addresses future research directions. Finally, Section 10 concludes the paper.

2 System Chip Test Challenges

In this section, the main challenges of testing system chips are analyzed and compared to the traditional chip test ap-

proach.

Even though the design process in core-based system chips is conceptually analogous to the one used in traditional chip design, but the manufacturing test processes in both cases are quite different. In the traditional system-on-board approach, chip manufacturing and testing are performed by the component provider, prior to PCB assembly and test done by the system integrator, as shown in Figure (1-a).

Whereas in a core-based system chip, the reusable cores are designed individually first by the component provider. Next, the system integrator designs the User Defined Logic (UDL) and assembles the predesigned cores. Only then, the manufacturing and test steps are performed. This is done for the whole system chip, as shown in Figure (1-b).

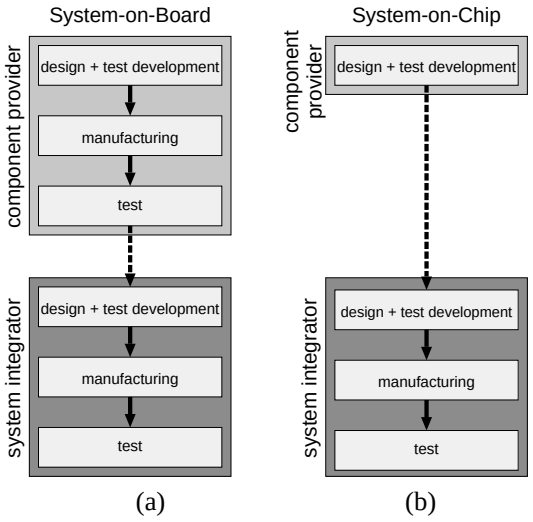

Figure 1: System-on-Board (a) vs. System-on-Chip (b) trajectory.

2.1 Core Level Test

A core is typically the hardware description of today's standard ICs, e.g., DSP, RISC processor, or DRAM core. Even though, a given core is not tested individually as in standard ICs, and instead is tested as a part of the overall system chip by the system integrator, but still, the preparation of a core internal test, i.e. the test development, is often done by the core provider, as in Figure (1-b). Because, the system integrator in most cases, except for soft cores, has very limited knowledge about the structural content of the adopted core, and hence deals with it as a black box. Therefore he/she cannot develop the necessary test for it. This is especially true if a core is a hard one or is an encrypted Intellectual Property block. This necessitates that the core provider develops the core test, i.e., the DfT structures and the corresponding test patterns, and delivers it with the core. Another core provider function is to determine the internal test requirements of the core without knowing the system chip environment and possibly even the target process. For instance, which test method needs to be adopted (e.g., BIST, scan, I_{DDQ}, functional test), what type of fault(s) (e.g., static, dynamic, parametric) to target and what level of fault coverage is desired. In the traditional approach, the overall chip test method and the desired fault coverage are predetermined. Hence, the designer incorporates the requirements during test development. But in system chips, a core provider often does not have enough information about the target applications of his component and their quality requirements. Hence, the provided quality level might or might not be adequate. If the coverage is too low, the quality level of the system chip is put at risk, and if it is too high the test cost may become prohibitive (e.g., test time, performance, area, power). Furthermore, different processes have different defect densities and distributions.

The core internal test developed by a core provider need to be adequately described, ported and ready for plug and play, i.e., for interoperability, with the system chip test. For an internal test to accompany its corresponding core and be interoperable, it needs to be described in an commonly accepted, i.e., standard, format. Such a standard format is currently being developed by IEEE P1500 and referred to as standardization of a core test description language (cf. Section 8).

2.2 Test Access

Another key difference between the traditional approaches and the ones for system chip is the accessibility to component peripheries, i.e., accessing primary input/outputs of chips and cores, respectively. With a system-on-board, direct physical access to chip peripheries, i.e., pins, is typically available to use probing during manufacturing test; whereas for cores, which are often deeply embedded in a system chip, direct physical access to its peripheries is not available by default, hence, an electronic access mechanism is needed. This access mechanism requires additional logic, such as a wrapper around the core and wiring, such as a test access mechanism to connect core peripheries to the test sources and sinks defined in Section 4. The wrapper performs switching between normal mode and the test mode(s) and the wiring is meant to connect the wrapper which surrounds the core to the test source and sink. The wrapper can also be utilized for core isolation. Typically, a core needs to be isolated from its surroundings in certain test modes. Core isolation is often required on the input side, the output side, or both. If it is on the input side, it can put the core into a safe-state by protecting the core under test from external interference (from preceding cores or UDL). On the output side, it protects the inputs of the superseding blocks (cores or UDL) from undesired values (e.g., random patterns applied to tri-state buffers

creating bus conflicts). Details about the necessary test circuitries are described in Sections 3 to 7 of this paper.

2.3 System Chip Level Test

One of the major challenges in the system chip realization process is the integration and coordination of the on-chip test and diagnosis capabilities. If compared to the conventional scheme, the system chip test requirements are far more complex than the PCB assembly test, which for instance in digital chips consists of interconnect and pin toggling tests. The system chip test is a single composite test. This test is comprised of the individual core tests of each core, the UDL test, and the test of their interconnects. As discussed earlier, each individual core or UDL test may involve surrounding components. Certain peripheral constraints (e.g., safe mode, low power mode, bypass mode) are often required. This necessitates access and isolation modes. In addition to the test integration and interdependence issues, the system chip composite test requires adequate test scheduling. This is needed to meet a number of chip-level requirements, such as total test time, power dissipation, area overhead, etc [2]. Also, test scheduling is necessary to run intra-core and inter-core tests in certain order not to impact the initialization and final contents of individual cores. With the above scheduling constraints the schedule of the composite system chip test is created.

In addition to the above differences between testing traditional chips and system chips, we have to note that system chips do also have the typical testing challenges of the very deep-submicron chips, such as defect/fault coverage, overall test cost, and time-to-market.

3 Conceptual Architecture

This section presents a conceptual architecture for testing embedded-core based system chips. The architecture consists of three structural elements, as depicted in Figure 2.

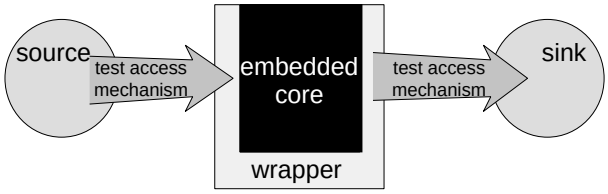

Figure 2: Overview of the three elements in an embedded-core test approach: (1) test pattern source and sink, (2) test access mechanism, and (3) core test wrapper.

1. *Test pattern source and sink.*
   The test pattern source generates the test stimuli for the embedded core, and the test pattern sink compares the response(s) to the expected response(s).

2. *Test access mechanism.*
   The test access mechanism transports test patterns. It can be used for on-chip transport of test stimuli from a test pattern source to the core-under-test, and for transport of test responses from the core-under-test to a test pattern sink.

3. *Core test wrapper.*
   The core test wrapper forms the interface between the embedded core and its environment. It connects the terminals of the embedded core to the rest of the IC and to the test access mechanism.

All three elements can be implemented in various ways, such that a whole palette of possible approaches for testing embedded cores emerges. In the subsequent sections we review the various alternatives and classify current approaches.

4 Test Pattern Source and Sink

The test pattern source generates the test stimuli for the embedded core. The test pattern sink compares the response(s) to the expected response(s). Test pattern source as well as sink can be implemented either off-chip by external Automatic Test Equipment (ATE), on-chip by Built-In Self-Test (or Embedded ATE), or as a combination of both. Source and sink do not need to be of the same type, e.g., the source of an embedded core can be implemented off-chip, while the sink of the same core is implemented on-chip. The choice for a certain type of source or sink is determined by the following.

1. The type of circuitry in the core.
2. The type of pre-defined tests that come with the core
3. Quality and Cost considerations.

In the sequel of this section we address each of these issues briefly.

We distinguish three main types of circuitry used in system chips today: (1) logic, (2) memory, and (3) analog and mixed-signal. Simple cores consist of one circuitry type only; complex cores consist of multiple simple cores, possibly of different circuitry type. These three types of circuitry exhibit different defect behavior and hence their tests are quite different in nature [3, 4]. This requires also different types of sources to generate the stimuli and sinks to compare the responses. ATE systems as well as BIST schemes for logic, memory, and analog are traditionally quite different. With the advent of system chips, which often incorporate various types of circuitry into one IC, we see that both

ATE vendors, as well as providers of BIST solutions, move towards integrating their traditionally separate solutions for logic, memory, and analog into combined product offerings.

The variety in types of core tests is much larger than the three circuitry types listed above. Tests can not only be classified by the type of circuit they test (logic vs. memory vs. analog), but also by the type of measurement they require (voltage vs. current), by the way they are generated (functional vs. structural), by the amount of core-internal adaptation they require (scan vs. test points), etc.

Examples of tests based on current measurements are I_{DDQ} and I_{DDT} [5], measuring quiescent and transient currents respectively. Current measurements can be done both by on-chip sinks (on-chip current monitors [6]) as well as off-chip sinks (off-chip current monitors at the load board or the PMU unit of an ATE system).

For random logic cores to be tested by an on-chip source requires these cores to be prepared for BIST [7, 8], but do not contain a pattern generator and response compacter. This means that scan and test points have been added, as well as proper DfT to prevent internal unknown values. By preparing a core for BIST, but not including the pattern generator and response compacter, the core user is provided with the option to make trade-offs between dedicated pattern generators and response compacters per core, versus a shared pattern generator and response compacter which serve multiple cores. The on-chip test resources are often described indirectly, e.g., by the primitive polynomial which characterizes the LFSR that generates the pseudo-random test patterns that achieve the quoted fault coverage [3]. Such cores, although prepared for an on-chip test pattern source, can also be served by an off-chip source.

Cores which come with functional tests and/or ATPG-generated deterministic tests in general do not have regular test pattern sets, despite the amount of functionality knowledge or sophisticated algorithms required for their generation. An on-chip source for irregular test patterns could of course be implemented as a large ROM, but this leads to prohibitive silicon area costs. Research is going on to generate irregular deterministic test patterns on-chip at acceptable area costs [9]. One line of thought is that this should be possible by allowing the on-chip source to generate a large test set, in which the required test pattern set is guaranteed to be contained. However, such on-chip sources are still in a research phase, and hence many resort to off-chip sources for cores if they require an irregular deterministic set of stimuli.

Off-chip sources and sinks often require quite large capital investment, and, because they are built using yesterday's microelectronics technology they suffer from two major problems: Firstly, as described by the 1997 National Technology Roadmap for Semiconductors, the increasing IC speeds require increasing accuracy for resolution of timing signals at the IC pins. However, while tester accuracy has improved at a rate of 12% per year, IC speeds have improved at 30% per year. This growing gap leads to reduced ability to properly identify bad chips, leading to major yield losses, and cost increases. Secondly, due to demands for higher speed, it is becoming increasingly difficult for the off-chip ATE to keep up with the very high frequencies needed to sufficiently test the performance related defects of today's ICs. In addition to these test quality related issues, the increased pin count and the mixing of diverse technologies in system chips will cause the ATE cost to rise towards US$ 20M, according to the 1997 NTRS. The above problems with the quality and cost of external ATE will only be accelerated by high-speeds, high-density, and mixed-technology system chips, thereby rendering external ATE unacceptably inaccurate and prohibitively expensive.

The type of circuitry of a certain core and the type of predefined tests that come with the core determine which implementation options are left open for test pattern source and sink. The actual choice for a particular source or sink is in general determined by quality and cost considerations. On-chip sources and sinks provide better accuracy and performance related defect coverage, but at the same time increase the silicon area and hence might reduce manufacturing yield. In theory all kinds of test patterns can be generated on-chip, but in practice only algorithmic patterns, such as the regular patterns for memory testing [3] functional patterns for analog core testing [10], or pseudo-random patterns for random logic testing [3] can be generated on-chip without requiring an excessive amount of silicon area. On-chip sources and sinks in most cases still need some form of off-chip ATE, e.g., for initialization or final comparison of a signature, and hence do not abandon those costs completely. The required quality and associated cost assessment is different for every individual case.

5 Test Access Mechanism

The test access mechanism takes care of on-chip test pattern transport. It can be used (1) to transport test stimuli from the test pattern source to the core-under-test, and (2) to transport test responses from the core-under-test to the test pattern sink. The test access mechanism is by definition implemented on-chip. This section gives an overview of the various implementation possibilities for a test access mechanism. Although for one core often the same type of test access mechanism is used for both stimulus as well as response transportation, this is not required and various combinations may co-exist.

Designing a test access mechanism involves making a trade-

off between the transport capacity (*bandwidth*) of the mechanism and the test application cost it induces. The bandwidth is limited by the bandwidth of source and sink and the amount of silicon area one wants to spend on the test access mechanism itself. A wider test access mechanism provides more bandwidth at the cost of more wiring area. A wide mechanism does not make much sense if the test pattern source is formed by an external ATE and the IC has only a few pins. The test time is a resultant of the test data volume of the individual cores and the bandwidth of the test access mechanism. How expensive test time per time unit actually is, depends on the type of source and sink. There is quite a wide range of external test equipment with dito associated test costs, and these again differ from the test application time costs of BIST.

When implementing a core test access mechanism, we have the following options.

- A test access mechanism can either reuse existing functionality to transport test patterns or be formed by dedicated test access hardware.
- A test access mechanism can either go through other modules on the IC, including other cores, or pass around those other modules.
- One can either have an independent access mechanism per core, or share an access mechanism with multiple cores.
- A test access mechanism can either be a plain signal transport medium, or may contain certain intelligent test control functions.

In the sequel of the section, several proposed and currently used test access mechanisms for testing core-based system chips are described. The techniques are based on Macro Test [11], transparency of cores [12, 13], reusing the system bus [14], multiplexed access [15], a dedicated test bus [16, 17], Boundary Scan [18, 19] and their partial variants [20, 21], test scheduler [2] and TESTRAIL [22].

Macro Test is a generic approach for testing embedded modules. It was developed by Beenker et al. [23, 24], originally focused on defect-oriented, and hence separate, testing of (embedded) modules with different circuit structures, and has been used on many industrial ICs [25]. With the advent of core-based design, the Macro Test approach proved also very useful for testing embedded cores [11]. In Macro Test, separation of tests into test protocols and test patterns plays a crucial role. Test Protocol Expansion translates core-level test protocols into IC-level test protocols by identifying suitable access paths. Test Protocol Scheduling optimizes the IC-level test time by intelligent scheduling of the various core tests. The flexible Macro Test approach supports all sorts of test access; it handles dedicated test access mechanisms, but can also take advantage of existing functionality where appropriate. Macro Test as such does not prescribe any particular test access mechanism.

An approach presented by Ghosh et al. [12, 13] is based on transparency of cores. It mandates that every core has a transparent mode, in which it propagates test data from its inputs to outputs without information loss. During the test of a particular core, transparent cores between the test pattern source and the inputs of the core-under-test form a test access mechanism, and likewise for transparent cores between the outputs of the core-under-test and the test pattern sink. Ghosh et al. consider it to be the core provider's responsibility to provide cores which are not only testable, but also transparent. A design-for-transparency technique is proposed which uses the functional description of a core to add minimal transparency hardware necessary. The core user is responsible for system-level test scheduling and test expansion from core terminals to IC pins. While lowering the test area and delay overheads compared to other techniques, the technique in [12] suffers from the drawback of relatively large test application times, due to the high transparency latency needed to propagate test data through each core. It also has the drawback that many cores, including legacy cores, may not have a functional description available. In [13], the approach is expanded through providing different versions of the core having different area overheads and transparency latencies. At the chip level, the method analyzes the topology of the system chip to select the core versions that meet the user's desired trade-off between area and test time objectives.

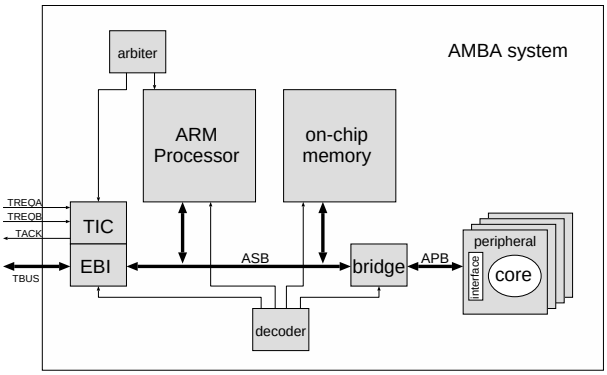

Figure 3: ARM's AMBA system.

An example of reusing existing functionality as test access mechanism is the test approach of ARM's Advanced Microcontroller Bus Architecture (AMBA) [14]. In this approach, the 32-bit system bus transfers test stimuli from IC pins via the External Bus Interface (EBI) to the core-under-test and test responses vice versa. In test mode, the Test Interface Controller (TIC) acts as bus master and controls the test mechanism (cf. Figure 3). Advantages of this approach

are (1) the low additional area costs of the test access mechanism, and (2) the relative simple test expansion because of the fixed relation between core terminals and IC pins. A disadvantage of the approach is that the fixed 32-bit bus does not allow to make trade-offs between area costs for the test access mechanism on one hand and test time on the other. Another disadvantage is that this approach, which works fine for the functionally-tested ARM core itself, dominates the overall IC test approach and makes it hard to integrate other test methodologies such as scan and BIST [26].

An obvious mechanism to make embedded cores testable from the IC pins makes the core-under-test directly and parallel accessible from the IC pins. This approach is commonly practiced for embedded memories, but also many block-based ASICs use this test access strategy [15]. Additional wires are connected to the core's terminals and multiplexed onto existing IC pins. The multiplexer control is coded as a dedicated test mode; in case of multiple embedded cores this leads to an equal number of test modes. Figure 4 gives a schematic view of the approach. The advantage of this approach are clear: as the embedded core can be tested as if it were a stand-alone device, the translation of core-level tests into IC-level tests is simple and straightforward. This also simplifies silicon debug and diagnosis. The disadvantage of this technique is that it is not very scalable. If the embedded core has more terminals than the IC has pins, the approach requires modifications. If a system IC contains many embedded cores, this approach leads to high area costs for connecting and multiplexing all core terminals to IC pins, and the control circuitry for these multiplexers will grow more complex.

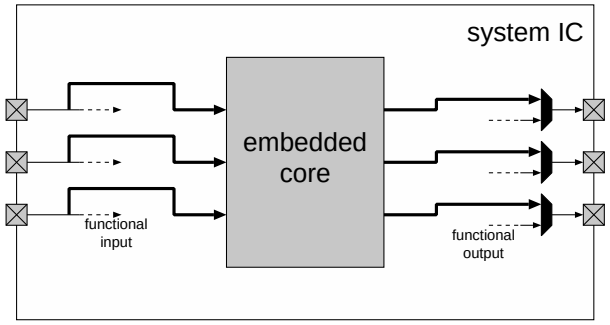

Figure 4: Multiplexed direct parallel access.

The test access mechanism as proposed by Varma & Bhatia [16, 17] also accesses the core-under-test directly from the IC pins, but provides the option to share one access mechanism with multiple cores. Such a shared access structure is called a *test bus*. Per IC, there are one or more test buses of varying width. Multiple cores can be connected to one tri-stateable test bus; all cores on the same bus are tri-stated, except for the core-under-test. This approach allows the system IC integrator to choose his optimal mix of number and width of test buses and number of cores per test bus and hence provides the opportunity to trade off silicon area for test time. Although presented as a dedicated test access mechanism, it is relatively easy in this approach to reuse existing on-chip buses as test bus implementation, as is done in ARM's AMBA system [14]. The main disadvantage of this approach is that per test bus, only one core can be connected at a time. This limits the possibilities to test multiple cores at the same time, which is sometimes desired (test time reduction through test scheduling, I_{DDQ} testing) or even required (testing interconnect wiring and glue logic in between cores).

The Boundary Scan Test (BST) approach, developed for board-level interconnect testing and standardized as IEEE 1149.1 [27], can be reused for testing embedded cores. In this approach, a serial scan chain is laid around the terminals of the core. Through this scan chain both the core itself, as well as the interconnect wiring and logic in between cores can be tested. The obvious benefit of this approach is that an existing test standard is reused. Many ICs are equipped with BST, often augmented with multiple private instructions which represent all kinds of test, debug, and emulation modes. If these ICs become cores in large system ICs, it seems logical to reuse the test infrastructure as provided by BST. A technical problem is that a scheme has to be developed to control multiple TAP controllers on one IC; various solution approaches have been proposed [18, 19]. A more serious drawback of this approach is that the BST standard defines a combined test control and test data access path of only one bit wide via TDI and TDO. Therefore, the BST approach does not allow to make a trade-off between bandwidth and test time. For small test pattern sets, the small bandwidth does not pose a problem. This is for example the case for board-level interconnect testing, for which BST was developed, or for cores equipped with BIST. However, for large scan-testable cores, providing many scan test patterns via only one serial access wire results in prohibitive long test times.

Touba & Pouya [20] have presented a variation on the Boundary Scan Test approach by using a partial boundary scan ring around the core. The ATPG techniques are used to find out which of a given set of stimuli can be justified from the (assumed) user-defined logic (UDL) in front of the core, such that the corresponding core inputs can be excluded from the partial boundary scan ring. In [21] the same authors even allow modification of the UDL such that the required test patterns can be generated.

In addition to its function as test data and control signal transportation medium, the Test Access Mechanism presented in Zorian [2] has an embedded intelligence to execute the system chip test in a predetermined schedule. The test sequencing of all the cores is embedded in a network of distributed

modular controllers, which is an integral part of the Test Access Mechanism. This network called, the Universal BIST Scheduler, is either customized to execute a predetermined BIST schedule for manufacturing test or is programmed on the fly through the JTAG port to execute the test of individual cores during silicon debug and diagnosis.

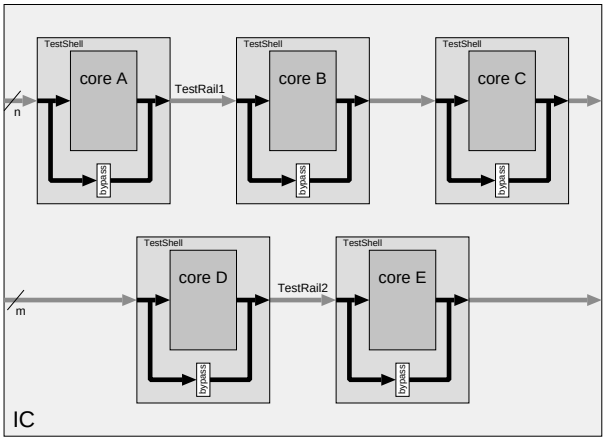

Figure 5: TESTRAIL example.

The TESTRAIL, as proposed by Marinissen et al. [22], tries to combine the strengths of both test bus and BST approach. A TESTRAIL forms a test access mechanism for one or more cores. Like with test bus, one can have one or more TESTRAILs of varying width per IC. This allows the core user to make trade offs between test time and silicon area. Like BST, multiple cores can be daisy-chained into one TESTRAIL. Per core, there is also a TESTRAIL bypass. This allows to core user to test each core sequentially or test multiple cores in parallel and hence to make trade offs between diagnostic resolution and test time. Figure 5 shows an example IC, containing five cores. Cores A, B, and C are connected in one TESTRAIL of width n, while cores D and E are connected in another TESTRAIL of width m. In [28] various TESTRAIL configurations in the context of scan-testable cores are analyzed w.r.t. test time. These configurations include (1) a wide private TESTRAIL per core, multiplexed with other TESTRAILs onto the IC pins, (2) all cores in one daisy-chained TESTRAIL with the usage of bypasses, and (3) a limited-size private TESTRAIL per core, such that all TESTRAILs have their own IC pins.

6 Core Test Wrapper

The core test wrapper forms the interface between the embedded core and its system chip environment. It connects the core terminals both to the rest of the IC, as well as to the test access mechanism. By definition, the core test wrapper is implemented on-chip.

The core test wrapper should have the following mandatory modes.

- *Normal operation* (i.e., non-test) mode of the core. In this mode, the core is connected to its system-IC environment and the wrapper is transparent.
- *Core test* mode. In this mode the test access mechanism is connected to the core, such that test stimuli can be applied at the core's inputs and responses can be observed at the core's outputs.
- *Interconnect test* mode. In this mode the test access mechanism is connected to the interconnect wiring and logic, such that test stimuli can be applied at the core's outputs and responses can be observed at the core's inputs.

Apart from these mandatory modes, a core test wrapper might have several optional modes, e.g., a *detach* mode to disconnect the core from its system chip environment and the test access mechanism, or a *bypass* mode for the Universal BIST Scheduler [2] and the TESTRAIL [22] test access mechanisms.

Depending on the implementation of the test access mechanism, some of the above modes may coincide. For example, if the test access mechanism uses existing functionality, normal operation and core test mode may coincide.

The core test wrapper connects the core terminals to the test access mechanism. The width of the test access mechanism does not necessarily correspond to the number of core terminals. The number of core terminals is typically determined by the function and application of the core. The width of the test access mechanism is also determined by the bandwidth of source and sink, as well as by the amount of silicon area the system-IC integrator wants to spend on the test access mechanism. If the number of core terminals is larger than the width of the test access mechanism, the *width adaptation* is done in the core test wrapper through serial-to-parallel conversion at the core inputs and parallel-to-serial conversion at the core outputs.

Pre-designed cores have their own internal clock distribution system. Different cores have different clock propagation delays, which might result in clock skew for inter-core communication. The system-IC designer should take care of this clock skew issue in the functional communication between cores. However, clock skew might also corrupt the data transfer over the test access mechanism, especially if this mechanism is shared by multiple cores. The core test wrapper is the best place to have provisions for clock skew prevention in the test access paths between the cores.

The *test collar* by Varma & Bhatia [17] and the TESTSHELL by Marinissen et al. [22] are examples of two similar core test wrappers. Both support the features as described above. Figure 6 depicts a small example of an IC with three scan-testable cores, using TESTRAIL and TESTSHELL. Table 1 lists the key parameters of the three cores. The TESTRAIL in this example has width three. The figure shows (1) the adaption in the various TESTSHELLs between the varying number of core terminals and the fixed number of TESTRAIL wires, (2) test access for both core test and interconnect, and (3) the implementation of the TESTRAIL bypass.

Core	# inputs	# outputs	# scan chains
A	-	5	3
B	5	3	2
C	3	-	2

Table 1: Parameters of the example IC.

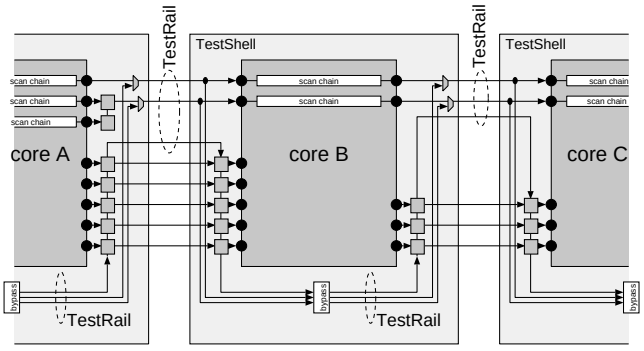

Figure 6: TESTRAIL and TESTSHELL example for scan-testable cores.

Of the entire core test architecture, the core test wrapper is the element which, if standardized, pre-eminently could contribute to interoperability of multiple cores of various source and type on one system chip. In Section 8.2 we describe the current status of the standardization of a core test wrapper by IEEE P1500, which contains exactly the features as described above.

7 Design Validation and Debug

While the use of pre-designed cores can reduce the product design cycle, and system-level integration improves system performance and costs, the productivity gains achievable may be significantly reduced by the challenges imposed by validation and debug of a system chip. Though each core of a system chip is pre-verified individually before system integration, validating the correct functionality and timing of the complete system chip, and debugging and diagnosing any possible error, can consume a significant portion of the design cycle. The main challenges for validating and debugging a system chip are: the heterogeneity of the system chip, the need to validate and debug software and hardware simultaneously, and the presence of deeply embedded cores with the associated accessibility problems as in testing.

A system chip needs to be validated against design and timing errors at various phases of its design cycle: specification and algorithmic validation, architectural validation, and finally prototype validation, the latter referring to validating the prototype or actual silicon of the system. Prototype-level system validation is the most accurate and fast, but debugging, diagnosing, and correcting errors at this level can be very costly, necessitating time consuming design iterations, and re-production of the prototype or silicon. Hence, several system validation techniques like co-simulation are being developed to facilitate validation of systems early in the design cycle. However, prototype-level validation and silicon debug remain indispensable, especially for systems that require real-time speed during validation, which is unattainable by current high-level validation methods, including co-simulation. In this paper, we describe several techniques for prototype validation, particularly focusing on silicon debug.

7.1 Prototype Validation

The most widely used prototype validation technique for conventional systems has been *emulation*, involving imitation of the target system or parts thereof by another system that typically uses Field-Programmable Gate Arrays (FPGAs) to implement the hardware parts of the target system, and processors to implement the software parts. While emulation allows nearly at-speed validation of the system (about 1 million cycles/sec), it can be very expensive. Moreover, though the availability of FPGA cores is on the rise, FPGA-based emulation may be inefficient and incompatible for a system chip integrator using hard cores, necessitating silicon debug.

Debug of system chips, consisting of programmable components like microprocessors and DSPs, requires the capability to monitor software execution on the programmable cores, and consequently access to the internal registers and buses of the deeply embedded processor cores. A problem with emulation-based validation and debug is the poor access it provides to the internals of processors. This problem can be overcome by an *In-Circuit Emulator (ICE)*, which is a box of hardware that can emulate the processor with the rest of the target system. The ICE can execute code in the target system's memory, and allows the user to perform breakpointing and other debug tasks. A disadvantage of ICE is the necessity to physically replace the actual processor by the ICE: errors due to electrical characteristics cannot be replicated when the ICE is used, and a deeply embedded CPU increases the cost for bringing internal signals out for the ICE to see.

A silicon debug technique, which allows debug of the actual processor in the target system chip, is the use of a *debug monitor*, which is a program residing in the memory of the system chip. The debug monitor controls program execution on the processor, communicating with the debug host system through a RS-232 or UART serial port. It can provide debug features like setting breakpoints, uploading data from target memory, and downloading application programs. A clear advantage of this approach is the ability to debug the processor cores, and the application software, in the presence of the other hardware cores. However, the need to have the debug monitor resident in the system chip's memory may entail significant extra cost, unless it is removed from the final product.

7.2 Embedding Debug in Cores

Many of the problems and costs associated with debugging system chips can be overcome by embedding appropriate debug capabilities in the cores. While standard ICE can be directly plugged to a processor in a system-on-board, the same cannot be done for a processor core in a system chip, incurring extra costs to access the deeply embedded core. In several processor and DSP cores, in-circuit emulation (ICE) is built-in the core itself, providing a "virtual ICE", allowing interfacing between a source-level debugger/emulator and the core embedded in a system chip. A few examples of embedded debug capabilities in processor and DSP cores are: EmbeddedICE in ARM 7TDMI processor core from ARM, ScanICE in the MiniRISC and OakDSP cores from LSI Logic, and the debug module in ColdFire microprocessor cores from Motorola.

We will explain the embedded debug methodology with a brief description of the embedded ICE capability of the ARM 7TDMI processor core [29, 30]. Figure 7 shows the ARM processor core, along with the embedded ICE debug architecture. The existing boundary scan chain (chain 0) that is around the core for production test is reused for debug, along with a new scan chain on the data bus (chain 1), an EmbeddedICE macro cell and associated scan chain (chain 2), and an extended TAP controller. The existing JTAG port needed for production test is reused for accessing the core during silicon debug. The debug architecture provides debug capabilities like real time breakpoints (on instruction fetches), watchpoints (on data loads and stores), single stepping, full control of the ARM processor core, and full access to the other parts of the system chip.

While scan chain 0 provides control to the address and data buses as well as the control registers in the processor core, the shorter scan chain 1 allows instructions and data to be inserted into the core without having to shift data around the entire periphery of the core. The EmbeddedICE macro cell resides on-chip with the processor core, and is used to enforce breakpoints and watchpoints. It contains breakpoint and watchpoint registers which can be programmed through the associated scan chain 2. The macro cell compares its register values with the address, data, and control values in chain 0, and halts execution of the processor core if a match occurs. For example, the macro cell may be programmed to halt processor execution when an instruction is loaded from a specified address, or a specified data value is stored to a given location. The processor core then enters a debug state which allows the internal system state, including registers and memory content, to be examined through the JTAG interface. The system state can also be changed if desired, like scanning in new instructions into the pipeline using scan chain 1. The processor core can be restarted once debugging is over.

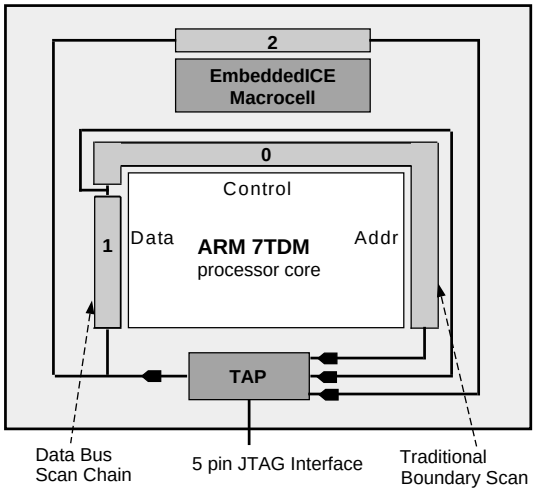

Figure 7: The ARM 7TDMI debug architecture, using the ARM EmbeddedICE macro cell, scan chains, and JTAG TAP controller.

Besides the system chip containing the core with embedded ICE, a complete debug system typically consist of a debug host running the debugger software, and an external interface unit, like the EmbeddedICE Interface box from ARM, which converts between the standard debugger protocol of the host and the protocol required by the JTAG interface.

The embedded debug capabilities in the ColdFire processor cores from Motorola provide support for real-time embedded system applications, besides the emulation debug support described above. For real-time systems, the processor cannot be halted during debugging, but must continue to operate concurrently with the debug module. To facilitate this, concurrent execution of the processor and debug module is allowed. Also, real-time trace support is provided, which allows tracing of the dynamic execution path of the application code on the processor core.

Besides the JTAG-based debug architectures described above, EJTAG, an Enhanced JTAG architecture, has been developed to support debugging of embedded MIPS cores in

system chips. The processor core can access external memory on the processor probe in a serial way through the JTAG pins, eliminating the need to use on-chip memory for debug routines. The processor core registers and memory can be accessed during processor execution through the optional EJTAG DMA master functionality via the on-chip bus system. In contrast to other JTAG-based debug units like EmbeddedICE from ARM and ScanICE from LSI Logic, the EJTAG implementation does not make use of test scan chains.

8 Industry-Wide Efforts

8.1 VSI Alliance

The *Virtual Socket Interface* (VSI) Alliance [31] is a business alliance of companies which tries to identify and/or define interface standards for design reuse of *virtual components*, the VSI term for embedded cores. Currently, over 200 companies from all segments of the semiconductor industry are VSI member. The VSI Alliance has the following seven Development Working Groups (DWGs): Implementation Verification, IP Protection, Manufacturing-Related Test, Mixed Signal, On-Chip Bus, System-Level Design, and Virtual Component Transfer. Two DWGs which relate to the scope of this paper are the Manufacturing-Related Test DWG and the Implementation Verification DWG. In the sequel of this section we describe their scope and planned deliverables.

The Manufacturing-Related Test DWG has around 20 active members, representing EDA companies, core providers, core users, and ATE companies. The DWG's scope includes the following topics.

- Virtual Component test
- Test access
- Test isolation
- Interconnect test
- Shadow logic test
- Test integration
- Test logic integration
- Failure identification

The Manufacturing-Related Test DWG has planned for the following deliverables.

1. Test data interchange specifications.
   Test vector formats (e.g., VCD, STIL), test structures, and interface specifications.
2. Guidelines for system chip integrators.
   Specification of system chip test architecture and test guidelines, testing of UDL and interconnect, system-level test, test integration, and debug.
3. Guidelines for VC providers.
   Guidelines on test methods, DfT, test isolation, test access, and debug.
4. Guidelines for mergeable IP providers.
   Guidelines on test methods and DfT rules.

Whenever possible, the VSI Alliance tries to leverage existing or emerging standards. In the scope of the Manufacturing-Related Test DWG, these are IEEE P1450 (better known as STIL, *Standard Test Interface Language* [32]) and IEEE P1500.

The Implementation Verification DWG is chartered with identifying the data representation standards which are required to complete the implementation and verification of soft, firm, and hard VCs itself, as well as the system chip design containing the virtual component. These verification tasks include functional, timing, and physical verification.

8.2 IEEE P1500

The IEEE P1500 Working Group [33] pursues a standard for testing embedded cores. The need for standardization arises because in core-based design and test, two parties are involved: the core provider and the core user. IEEE P1500 tries to define a standard which facilitates the test interoperability of cores from different sources, in order to improve the efficiency of both core provider and core user.

P1500 does not standardize the core's internal test methods or DfT, nor system-IC test integration and optimization issues, such as the type of test pattern source, sink, or transport mechanism. The P1500 standardization encompasses two main topics.

1. A standard language capable of expressing all test-related information to be transferred from core provider to core user.
2. A standardized, but configurable core test wrapper, which allows easy integration of the core into a system chip design.

The P1500 Working Group has two Task Forces which work on these respective topics.

8.2.1 P1500 History and Activities

In September 1995, the IEEE Test Technology Technical Committee (TTTC) initiated a Technical Activity Committee (TAC) in the field of test and DfT for core-based system ICs. After organizing several awareness meetings in conjunction with major design and test conferences, the TAC evolved into a Standardization Working Group, of which the PAR was approved by the IEEE Standards Activities Board in June 1997 [34].

IEEE P1500 has active participation from leading companies in industry segments such as systems organizations, EDA vendors, core providers, IC manufacturers, and ATE vendors, as well as from the research community. The P1500 Working Group maintains relations with the VSI Alliance, in particular with its Manufacturing-Related Test DWG, as well as with other standardization Working Groups in the test domain, such as IEEE 1149.x (Boundary Scan Test) and IEEE P1450 (STIL).

Regular P1500 Working Group meetings are organized, often in conjunction with major conferences. In 1998, five of those meetings have taken place: DATE, VTS, DAC, an additional meeting in September, and ITC. These meetings are open for everyone interested. P1500 maintains an Internet site [33] and an e-mail reflector (`coretest@computer.org`).

8.2.2 Core Test Description Language Task Force

The Core Test Description Language (CTDL) Task Force of P1500 focuses on defining a standard language in which all test-related information to be transferred from core provider to core user can be expressed. This includes the following.

- Test methods.
- Test modes and corresponding test protocols.
- Test pattern data, e.g., lists of test vectors, march algorithms for memories, primitive polynomials for BIST-ready logic cores.
- Data on fault models and fault coverage.
- Information of core-internal design-for-test hardware, such as number, length, and order of scan chains, BIST, etc.
- Information on core-internal design-for-debug hardware, such as scan chains, hardware breakpoints, etc.
- Diagnostic information, e.g., physical location of probe points, etc.

Obviously, not all core providers will make all this data available for their products. Therefore, some data is mandatory, whereas other is optional.

The expressing power of and concepts behind such a language are thought to be crucial, whereas the actual syntax of the language is considered to be less important. Therefore, the current line of thinking is that this Task Force might set the requirements for extension of another standard language, IEEE P1450 (STIL). Currently, STIL focuses at describing test patterns and waveforms for ICs [35], and, given the bullet list above, it will need several extensions to be suitable as Core Test Description Language.

8.2.3 Scalable Architecture Task Force

The Scalable Architecture Task Force of P1500 tries to define a uniform but flexible hardware interface between an embedded core and its environment, capable of delivering pre-defined test patterns to and from the embedded core. The hardware interface should be standardized, in order to guarantee easy integration ('plug-n-play') and inter-operability of a core w.r.t. testing. On the other hand it should be flexible, to allow the core provider but especially the core user to make design trade-offs between test quality, test development time, silicon area, and performance impact.

The standardization work focuses on the definition of a core test wrapper, although aspects of the test access mechanism are taken into account as well. The current line of thinking is that the core test wrapper as standardized by P1500 has the following features.

- Multiple modes, including *normal*, *core test*, *interconnect test*, and *bypass* mode.
- May connect any number of core terminals to a test access mechanism of any width. Width adaptations, if necessary, are done through serialization.
- The core test wrapper has provisions for clock skew prevention on the test access paths between cores.
- The various modes of the core test wrapper are controlled via a serial control path, containing a dual shift/update register, and can optionally be extended to control core-internal test modes as well.

The standard will enable core providers to deliver P1500 compliant cores, for which the core user can count on a certain time-to-market benefit, because it guarantees relatively easy integration into any system chip environment. The standard will also enable EDA vendors to build tools that (1) turn non P1500-compliant cores into compliant cores, and (2) verify that a certain core is indeed P1500 compliant.

9 Future Challenges

In the preceding sections, we have described proposed methodologies and current industrial practices in testing and debugging core-based system chips. In this section, we analyze the effects of evolving design methodologies and the aggressive semiconductor technologies on testing system chips. We will also address problems due to limitations of test equipments. We describe the new test challenges imposed on system chips, and future research directions.

9.1 Effect of Design Methodologies

The attributes that make system chips using IP cores an attractive system design methodology are the very ones that make testing and debugging systems-on-chips a complex challenge: design reuse, heterogeneity, reconfigurability, and customizability. The core-based system chip design methodology facilitates design reuse of cores from various sources, and integration of heterogeneous components: from processor cores to complex multimedia cores, digital and analog components, synchronous and asynchronous interfaces, and even combination of electrical and mechanical components. While design reuse has the potential for significantly improving design productivity, it poses serious challenges to testing the system chip, with a wide variation in test strategies, test structures, and test requirements employed by the heterogeneous components. While development of test standards by the IEEE P1500 WG and the VSIA, as described in Section 8, are necessary steps to ease the tasks of chip-level test access and test integration, new test methodologies need to be developed to address the following issues.

- *Analog/Mixed Signal Cores and System Chips*.
  Several applications, like wireless telecommunication products, demand mixed-signal systems, consisting of digital, radio-frequency (RF), analog, and mixed-signal components. As system integration technologies advance, system chips consisting of RF, analog, and micro-electromechanical components, and cores containing new high performance and low power devices, will become common. Techniques need to be developed for testing such cores, and issues like test specification, access mechanisms, and isolation mechanisms need to be investigated, to achieve seamless integration with the evolving test standards for the digital counterparts.

- *Customizable and Reconfigurable Cores*.
  A major advantage of system design using IP cores and system-level integration is the configurability and customizability available to the system designer. In the system-on-chip design paradigm, the processor core, the associated peripheral units, and the underlying bus, memory and communication architectures, can all be configured and customized to best match the embedded system application, thereby giving tremendous cost and performance advantages over traditional system-on-board approaches. As tools and methodologies mature to enable the realization of configurable and customizable system chips, issues related to standardization of test and access mechanisms need to be addressed. For example, when the cores themselves can be customized by core users, how can test sets and test structures be pre-determined for reuse during system chip integration and test?

- *Soft Cores*.
  As core-based design reuse methodologies mature, and the reuse of non-processor functions and protocols, in multimedia and telecommunication systems, become more prevalent, the delivery and use of soft cores will increase significantly. Unlike hard cores, a soft core, which is typically a high-level description of the functionality, cannot be characterized for testability, and hence test reuse cannot be achieved. To enable delivery of testable soft cores, and test reuse, effective high-level testability analysis and design-for-testability techniques need to be developed.

- *Validation and Debug*.
  As software content in system chips increases, and complex buses and protocols are used to communicate and interface between on-chip components, debugging and diagnosing design errors will consume a significant portion of the design cycle time. As each core used in a system chip is pre-verified, the primary source of design errors would be in the interfaces and protocols between the cores. To make the system chip validation and debug problem more tractable, an interface-based system chip validation and debug methodology need to be developed, and validation wrappers and access mechanisms need to be defined, like in the case of testing for manufacturing defects. Also, similar to the EmbeddedICE debug architecture of the ARM processor core described in the previous section, debug structures can be embedded in other cores, enabling easy debug of system chips. The embedded debug structures should maximally reuse the embedded test structures, as in the case of the ARM core.

9.2 Effects of Deep Sub-Micron Technology

The 1997 National Technology Roadmap for Semiconductors (NTRS) predicts reusable core-based system chips using 100 nm technology, $>$2 GHz clock frequency, and 1.2 V

power supply by the year 2006. This will lead to growing effects of noise, due to increasing cross-coupling capacitances, inductances, and electro-magnetic fields. Recent studies show significant increases in signal delay and hazards due to cross-coupling capacitances between bus interconnects: the effects being most dramatic for wire lengths $\geq$10 mm, but becoming significant for wire lengths as low as 2 mm for 100 nm technology [36]. Also, the cores themselves may not be immune for chip-level noise; for instance, bus interconnects going over a core may affect the operation of the core. The above findings confirm the NTRS objectives of developing signal integrity tests for interconnects between cores, and to ensure the proper operation of cores in a system.

Several crosstalk extraction and analysis tools have been recently developed; while useful for design validation, they cannot be used for manufacturing testing, and the generation of the simulation vectors is not clear. Hence, proper attention must be given to the development of fault models, and test and diagnosis methodologies, for crosstalk and other noise, in buses and global interconnects connecting the cores of a system, as well as interference effects on the cores themselves. Soft errors, due to accumulation of system noise, needs to be considered. Also, since the effect of crosstalk faults will be most evident in high-frequency circuits, the test and diagnosis methodologies developed will be required to enable at-speed testing.

9.3 Test Equipment

As reported in Section 4, external ATE faces accuracy and performance limitation and is expected to become prohibitively expensive.

A common solution to the above problems can be attained by using self-testing methods for system chips, thereby allowing at-speed testing, and also reducing the need for external testing by ATE. This can be pursed by putting the external ATE functionality, or the speed-critical parts of it, on the system chip itself, thereby eliminating the speed-related problems of external ATE. The tradeoff between external ATE cost and the extra silicon cost of embedded ATE needs to be evaluated.

While BIST is emerging as a viable embedded ATE methodology for certain types of cores like embedded memory cores [3], random logic cores [3] and analog cores [10], self-testing methodologies need to be developed for complex cores like processor and DSP cores, and other types of cores, like RF, flash and MEM cores. On-chip resources may be reusable to generate self-tests for the other memory, logic and analog cores present in a system chip.

To summarize, there are significant test challenges imposed by system chip design methodologies, deep sub-micron technologies, and external ATE. These system chip test problems need to be addressed, so that the productivity gains of system chips are not diminished by rising testing costs, reduced product quality and yield losses.

10 Conclusion

Advances in IC process technology allow the integration onto a single IC of complete systems which, until recently, could only be implemented with multiple ICs on a board. In order to reduce the enormous amount of knowledge and time needed to development such system chips, designers are increasingly using pre-designed modules, so-called embedded cores. Two key challenges in effectively and efficiently using embedded cores are discussed in this paper, viz. manufacturing test and design validation and debug.

We presented a conceptual architecture for testing embedded cores with pre-defined tests. This architecture consists of three elements.

1. A test pattern *source* and *sink*. These can be implemented off-chip as well as on-chip. Both alternatives are currently used and have their specific advantages and disadvantages, especially w.r.t. silicon area and test time cost.

2. A *test access mechanism*. Various test access mechanisms have been described, using transparent paths through other cores, the system bus, a Boundary Scan-like approach, a dedicated test bus, and TESTRAIL.

3. A core test *wrapper*, which, in various modes, connects core terminals with either other IC modules or the sink and source, through the test access mechanism.

The challenges in design validation and debug are sketched.

We described the industry-wide efforts conducted by the VSI Alliance and the IEEE P1500 WG to come up with solutions for the presented challenges, which, if indeed adopted throughout the semiconductor industry, will stimulate interoperability of cores and hence help the industry forward. We also listed the future challenges, including the mixed-signal and soft cores, and the effects of deep sub-micron process technologies.

Acknowledgments

The authors thank the members of the IEEE P1500 Working Group for many stimulating discussions in this domain. Rudy Garcia of Schlumberger ATE, co-chair of VSIA's

Manufacturing-Related DWG, is kindly acknowledged for providing us with a recent status update. We thank Robert Arendsen and Maurice Lousberg of Philips for providing us with useful feedback on draft versions of this paper.

References

[1] Rajesh K. Gupta and Yervant Zorian. Introducing Core-Based System Design. *IEEE Design & Test of Computers*, 14(4):15–25, December 1997.

[2] Yervant Zorian. A Distributed BIST Control Scheme for Complex VLSI Devices. In *Proceedings IEEE VLSI Test Symposium*, pages 6–11, April 1993.

[3] V.D. Agrawal, C.J. Lin, P.W. Rutkowski, S. Wu, and Y. Zorian. Built-In Self-Test for Digital Integrated Circuits. *AT&T Technical Journal*, Vol. 73(No. 2):30, March 1994.

[4] Dhanendra Jani and John Acken. Synthesising processors. *Test - The European Industry Journal*, pages 17–20, April 1995.

[5] Manoj Sachdev, Peter Janssen, and Victor Zieren. Defect Detection with Transient Current Testing and its Potential for Deep Sub-micron CMOS ICs. In *Proceedings IEEE International Test Conference*, October 1998.

[6] Kenneth Wallquist, Alan Righter, and Charles Hawkins. A General Purpose I_{DDQ} Measurement Circuit. In *Proceedings IEEE International Test Conference*, pages 642–651, October 1993.

[7] Dwayne Burek Yervant Zorian and R. Chandramouli. A 2-Step Strategy tackles System-on-a-Chip Test. In *Digest of Papers of IEEE International Workshop on Testing Embedded Core-Based Systems*, pages 3.2–1–5, November 1997.

[8] Janusz Rajski, Tom Eberle, and Jerzy Tyszer. Modular Logic Built-In Self-Test for IP Cores. In *Digest of Papers of IEEE International Workshop on Testing Embedded Core-Based Systems*, pages 3.3–1–5, November 1997.

[9] Hans-Joachim Wunderlich and Gundolf Kiefer. Scan-Based BIST with Complete Fault Coverage and Low Hardware Overhead. In *Digest of Papers of IEEE European Test Workshop*, pages 60–64, June 1996.

[10] Stephen Sunter and Naveena Nagi. A Simplified Polynomial-Fitting Algorithm for DAC and ADC BIST. In *Proceedings IEEE International Test Conference*, pages 389–395, November 1997.

[11] Erik Jan Marinissen and Maurice Lousberg. Macro Test: A Liberal Test Approach for Embedded Reusable Cores. In *Digest of Papers of IEEE International Workshop on Testing Embedded Core-Based Systems*, pages 1.2–1–9, November 1997.

[12] Indradeep Ghosh, Niraj K. Jha, and Sujit Dey. A Low Overhead Design for Testability and Test Generation Technique for Core-Based Systems. In *Proceedings IEEE International Test Conference*, pages 50–59, November 1997.

[13] Indradeep Ghosh, Sujit Dey, and Niraj K. Jha. A Fast and Low Cost Testing Technique for Core-based System-on-Chip. In *Proceedings Design Automation Conference*, pages 542–547, June 1998.

[14] Bruce Mathewson. Core Provider's Test Experience. http://grouper.ieee.org/groups/1500/pastmeetings.html#dac98. Presentation at IEEE P1500 Working Group Meeting, Sunnyvale, CA, June 1998.

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

[16] Prab Varma. Evolving Strategies for Testing Systems on Silicon. *Integrated System Design- Virtual Chip Design Supplement*, September 1997.

[17] Prab Varma and Sandeep Bhatia. A Structured Test Re-Use Methodology for Systems on Silicon. In *Digest of Papers of IEEE International Workshop on Testing Embedded Core-Based Systems*, pages 3.1–1–8, November 1997.

[18] Lee Whetsel. An IEEE 1149.1 Based Test Access Architecture for ICs with Embedded Cores. In *Proceedings IEEE International Test Conference*, pages 69–78, November 1997.

[19] Debashis Bhattacharya. Hierarchical Test Access Architecture for Embedded Cores in an Integrated Circuit. In *Proceedings IEEE VLSI Test Symposium*, pages 8–14, April 1998.

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

[21] Bahram Pouya and Nur Touba. Modifying User-Defined Logic for Test Access to Embedded Cores. In *Proceedings IEEE International Test Conference*, pages 60–68, November 1997.

[22] Erik Jan Marinissen et al. A Structured and Scalable Mechanism for Test Access to Embedded Reusable Cores. In *Proceedings IEEE International Test Conference*, October 1998.

[23] Frans Beenker, Karel van Eerdewijk, Robert Gerritsen, Frank Peacock, and Max van der Star. Macro Testing: Unifying IC and Board Test. *IEEE Design & Test of Computers*, Vol. 3(No. 4):26–32, December 1986.

[24] Frans Beenker, Ben Bennetts, and Loek Thijssen. *Testability Concepts for Digital ICs - The Macro Test Approach*, volume 3 of *Frontiers in Electronics Testing*. Kluwer Academic Publishers, Boston, 1995.

[25] Frank Bouwman, Steven Oostdijk, Rudi Stans, Ben Bennetts, and Frans Beenker. Macro Testability; The Results of Production Device Applications. In *Proceedings IEEE International Test Conference*, pages 232–241, September 1992.

[26] Chris Feige, J. ten Pierick, Clemens Wouters, Ronald Tangelder, and Hans Kerkhoff. Integration of the Scan-Test Method into an Architecture Specific Core-Test Approach. In *Digest of Papers of IEEE European Test Workshop*, May 1998.

[27] IEEE Computer Society. *IEEE Standard Test Access Port and Boundary-Scan Architecture - IEEE Std 1149.1-1990*. IEEE, New York, 1990.

[28] Joep Aerts and Erik Jan Marinissen. Scan Chain Design for Test Time Reduction in Core-Based ICs. In *Proceedings IEEE International Test Conference*, October 1998.

[29] Advanced RISC Machines Ltd. *The ARM7TDMI Debug Architecture*, December 1995. ARM DAI 0028A, http://www.arm.com/Documentation/AppNotes/Apps28vA.

[30] Liam Goudge. Debugging Embedded Systems. http://www.arm.com/Documentation/WhitePapers/DebugEmbSys.

[31] VSI Alliance Web Site. http://www.vsi.org/.

[32] IEEE P1450 Web Site. http://grouper.ieee.org/groups/1450/.

[33] IEEE P1500 Web Site. http://grouper.ieee.org/groups/1500/.

[34] Yervant Zorian. Test Requirements for Embedded Core-Based Systems and IEEE P1500. In *Proceedings IEEE International Test Conference*, pages 191–199, November 1997.

[35] Tony Taylor and Gregory Maston. Standard Test Interface Language (STIL): A New Language for Patterns and Waveforms. In *Proceedings IEEE International Test Conference*, pages 565–570, November 1996.

[36] Petra Nordholz et al. Signal Integrity Problems in Deep Submicron Arising from Interconnects between Cores. In *Proceedings IEEE VLSI Test Symposium*, pages 28–33, April 1998.