Paper 12.1 at
International Test Conference,
Washington, D.C., October 1998

A Structured and Scalable Mechanism for Test Access to Embedded Reusable Cores

Erik Jan Marinissen[1a] Robert Arendsen[2a] Gerard Bos[2b]
Hans Dingemanse[3] Maurice Lousberg[1b] Clemens Wouters[2c]

[1]Philips Research Laboratories,
[a]VLSI Design Automation & Test,
[b]Electronic Design & Tools,
Prof. Holstlaan 4,
5656 AA Eindhoven, The Netherlands
{marinis,lousberg}@natlab.research.philips.com

[2]Philips Semiconductors,
ASIC Service Group,
[a]ESTC / [b]Microtel / [c]AMDC,
Prof. Holstlaan 4,
5656 AA Eindhoven, The Netherlands
{arendsen,bosg,cwouters}@natlab.research.philips.com

[3]Philips Semiconductors,
Consumer Systems TPE,
Infotainment Systems,
Gerstweg 2,
6534 AE Nijmegen, The Netherlands
Hans.Dingemanse@nym.sc.philips.com

Abstract

The main objective of core-based IC design is improvement of design efficiency and time-to-market. In order to prevent test development from becoming the bottleneck in the entire development trajectory, reuse of pre-computed tests for the reusable pre-designed cores is mandatory. The core user is responsible for translating the test at core level into a test at chip level. A standardized test access mechanism eases this task, therefore contributing to the *plug-n-play* character of core-based design. This paper presents the concept of a structured test access mechanism for embedded cores. Reusable IP modules are wrapped in a TESTSHELL. Test data access from chip pins to TESTSHELL and vice versa is provided by the TESTRAIL, while the operation of the TESTSHELL is controlled by a dedicated test control mechanism (TCM). Both TESTRAIL as well as TCM are standardized, but open for extensions.

1 Introduction

Modern semiconductor process technologies and design tools enable the design of complex systems-on-chips, comprised of millions of transistors. To efficiently use design resources and improve on time-to-market, there is a trend to embed reusable (parametrized) versions of large modules, the so-called *cores*. Examples of cores are CPUs, DSPs, PCI modules, MPEG and JPEG modules, and memories. Cores might come as *hard* (layout), *firm* (netlist), or *soft* (register-transfer level) descriptions. Core-based design divides the IC design community into two groups, viz. the *core providers* and the *core users*. In the entire product creation process for core-based designs we distinguish the following functions: (1) IP development, (2) core customization and packaging, and (3) core integration (see Figure 1). This new style of design takes place both within companies, as well as between companies. In the latter case, the core-providing company might want to protect its intellectual property rights (IPR) on the design of the core, either by legal or technical measures.

Core-based design is primarily focussed on design efficiency. However, if no attention is paid to the specific issues that are related to design-for-testability (DfT) and test generation for such modular, highly complex circuits, test may become the bottleneck in the entire IC development trajectory and hence the obstacle to actually cash in on the advantages of core-based design [1]. The obvious solution to this problem is to apply the same reuse paradigm that is used in IC design also in the test development. This means that together with the core's design description the core provider also delivers a pre-computed set of tests for the core. The core user's responsibility is to translate these pre-computed tests, that are described at the terminals of the embedded core, into chip-level tests, described at the IC pins. We refer to this activity as *test expansion*.

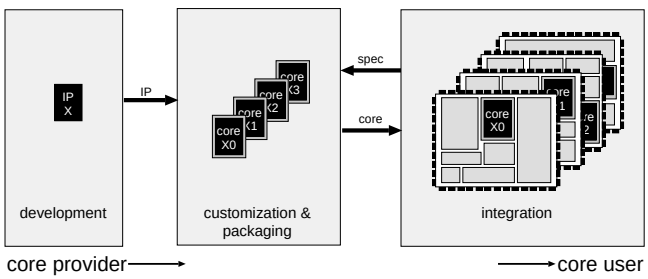

Figure 1: The product creation process of core-based design.

In general, test expansion is a difficult task. Boosten & Jacobs [2] proved that test expansion at *test protocol* [3] level is NP-hard. If insufficient access paths to the embedded core-under-test exist, test expansion is infeasible. Tool support is required to keep track of all the bookkeeping involved in test expansion. A test access mechanism, in the form of dedicated design-for-testability hardware that guarantees test access from the IC pins to the embedded core and vice versa, can alleviate the test expansion task enormously. In the case of multiple cores of different core providers on one IC, a standardized test access mechanism enables a unified test expansion for all cores.

This paper proposes a test access mechanism for embedded reusable cores. The test access mechanism supports a wide variety of core tests, including function test, scan test, memory test, built-in self test, I_{DDQ} test, etc. It does not rely on assumptions about the DfT inside the core-under-test, nor on DfT or transparent modes of other cores or design modules within the same IC. The mechanism supports *interconnect testing*, i.e., testing the wires and logic in between the various cores of a given IC design. The test access mechanism is scalable, such that trade-offs can be made between the bandwidth required for timely execution of the core tests and the silicon area costs involved. Next to its scalability, the proposed test access mechanism is structured, meaning that it could serve as the basis for a world-wide, standardized test access mechanism, such as is pursued by IEEE P1500 [4].

The sequel of this paper is organized as follows. Section 2 describes core testing based on sets of pre-computed tests and the requirements for any core test strategy that evolves from that. Section 3 describes prior work done in this field. Section 4 highlights the similarities and differences between board test and core test and shows that Boundary Scan Test (IEEE 1149.1) alone is insufficient to solve the core test issue. Section 5 presents the TESTSHELL, a small interface layer between the reusable IP module and its environment. Sections 6 and 7 describe the test access mechanism, called TESTRAIL, and the test control mechanism of the TESTSHELL respectively. Section 8 gives an application example for our method. Section 9 concludes the paper.

2 Core Test

2.1 Core test requirements

Core-based design aims at increasing design efficiency, i.e., with equal man power larger ICs should be designed faster. In order to prevent test development from becoming the bottleneck in the entire IC development trajectory, core test should be focused on speeding up test development for core-based designs. In our view, a sound core test strategy should fulfill the following requirements.

1. Reduce time-to-market by
   - *Reuse* of pre-computed tests as delivered with the core by the core provider.
   - Enable *easy integration* w.r.t. testing.
2. Support
   - *All types* of core tests, including function test, scan test, BIST, memory test, and I_{DDQ} test, as well as testing the interconnect wiring and logic in between cores.
   - *Hierarchy*. Today's cores often contain multiple modules of various circuit structures (random logic, SRAM, DRAM, etc.), which require different tests. And for the future an increased number of hierarchy levels can be expected ("today's ICs are tomorrow's cores").
   - *Automated expansion* of core-level tests into IC-level tests.
   - Provisions for *multiple clock* testing and *clock skew* prevention.
3. Cost-effective w.r.t. well-known cost factors such as silicon area, pins, test time, performance, power consumption, etc.

Note that the last requirement, i.e., cost-effectiveness, can be contradictory with the first requirement, i.e., short time-to-market. In our view, core-based design is driven by time-to-market motives, and hence this requirement should prevail. An area- or performance-driven design project should not choose a core-based design style in the first place. Of course an ideal core test strategy is one that provides the flexibility to make trade-offs between the various requirements.

2.2 Reuse of core-level tests

Where designers (re)invented the reuse paradigm to speed up IC design, it seems obvious to employ that same reuse paradigm to speed up IC test development. This implies that together with the core's design description the core provider should also deliver a set of core tests, defined at the core's input/output terminals. Just like the core design is embedded into an IC design 'as is' for reasons of design efficiency, the tests of the core should also be used 'as is'.

In general, development of high-quality tests requires a certain amount of design knowledge and often also involves design adaptations (design-for-test). Hence, it seems proper to place this task with the core provider, as he has in-depth knowledge of the core's design.

Reuse of core-level tests requires a trustful relation between core provider and core user, because the quality level of the core user's products will depend on the fault detection qualities of the core test as delivered by the core provider. However, this is not different in the design domain, where the functionality of the core user's products depends on the functionality of the core design as provided by the core provider. If such dependencies, especially between different companies, currently frighten us, then this is a sign that core-based design is only in its infancy yet and that the semiconductor industry still needs to get used to such inter-company relations.

2.3 Ease of integration

As time-to-market is the overriding goal of core-based design, the ease with which reusable cores can be embedded in an IC design is crucial. This is also true in the test domain. Even if a core comes with a pre-computed set of tests, and hence saves the core user the work of test development for that particular core, there is still work to be done by the core user. Typical tasks which can only be done by the core user, because they depend on the embedding of the core in a certain IC design, include the translation of core-level tests into IC-level tests and the scheduling of the execution order of the various core tests. Structured and scalable test access and test control mechanisms for embedded cores, capable of handling a variety of core designs and core tests, form an important means to ease the integration effort to be spent by the core user. The TESTSHELL and TESTRAIL as proposed in this paper are such mechanisms.

3 Prior Work

Various approaches for test access to embedded modules have been reported in the literature.

Already in the mid 1980s, Philips Research developed *Macro Test* [5]. Within Macro Test, a test is broken down into a *test protocol* and a list of *test patterns*. In order to reduce the computational complexity, translation of core-level tests into IC-level tests is done for test protocols [6, 3] instead of for full tests. The so-called *test protocol expansion* algorithms in the Macro Test tool suite are capable of identifying any mapping between core terminals and IC pins, including both combinational and sequential transparent access paths through other modules. Such a liberal definition of test protocol expansion is very flexible and can cope with any access scheme, but on the other hand provides very little guidance to the designer and test engineer in how to make an embedded core actually accessible.

Ghosh et al. propose a method in which test access to embedded cores is based on transparent paths through other cores and design modules [7]. In their view, every core should not only come with a set of pre-computed tests, but also with a set of transparent paths, capable of transporting test data through the core. If these paths do not exist 'naturally', the core provider should add design-for-test hardware to his core for the sole purpose of creating test access paths to other cores. This access method does not seem to address the issue of time-to-market and in many cases yields a superabundance of access paths and dito area usage.

A very obvious way to create test access to embedded modules is to connect the terminals of the embedded module directly to IC pins. In the case of multiple embedded modules, the IC often does not have sufficient pins to accommodate for all terminals, and hence multiplexing has to be applied, as described by Immaneni & Raman [8]. The principle behind this technique is that the IC has several test modes, and in every test mode, one of the embedded modules is accessible from the IC pins for testing. An apparent drawback of this technique is that it requires a relatively large amount of additional silicon area for multiplexers and wiring.

Varma & Bhatia describe a test access mechanism for embedded cores, named VisibleCores [9]. Their approach is based on two dedicated on-chip variable-width buses, one for transporting test control signals, and one for transporting test data signals. Embedded cores can be either connected or disconnected from the test buses. Test access from chip pins to embedded core and vice versa is achieved by connecting the core-under-test to the test data bus, and disconnecting all other cores. A disadvantage of this method is that only one core at a time can be connected to the test bus, while some tests involve multiple cores.

4 Board Test versus Core Test

There are strong similarities between ICs (*components*) on a printed circuit board and cores (*virtual components*) in an IC. Core-based design can be regarded as the next step in on-chip integration; what used to be components on a board are now cores in an IC.

The components at board level are assumed fault free, because they have been tested for manufacturing faults by their manufacturer. Hence, board test focuses on testing the *interconnects* between the components. Several techniques for doing this exist. One of the most effective and popular ones is *Boundary Scan Test*, defined as IEEE standard 1149.1 ('JTAG') [10, 11, 12]. The Boundary Scan Test standard defines the additional hardware for the IC, in order to solve the board interconnect test problem. An international standard was needed, because boards may have components

of different semiconductor manufacturers; a standardized approach ensures that the various components work together w.r.t. board test.

The Boundary Scan Test standard mandates a (boundary) scan chain along all input and output pins of the IC (see Figure 2). The length of this chain is variable, as it depends on the number of pins of the IC in question. The Boundary Scan hardware interfaces to the external world via a standardized set of pins, the so-called *Test Access Port* (TAP), consisting of TCK, TMS, TDI, TDO, and (optional) TRSTN. Its operation is controlled by a standardized on-chip controller, the so-called *TAP Controller*, which is implemented as a finite state machine.

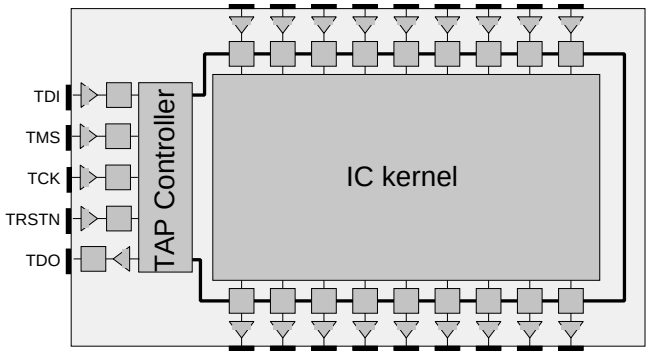

Figure 2: Overview of the Boundary Scan Test concept.

In spite of all similarities between board test and core test, there are two major differences between them.

- Whereas in board test the components are assumed to be tested by their manufacturer and hence fault free, this is not the case for the virtual components in core test. The core provider does not deliver hardware, but only a description thereof. Independent whether the provided core is hard, firm, or soft, it is merely a description, and hence not yet manufactured. This implies that it is impossible for the core provider to test its core for manufacturing faults. This can only be done once the IC containing the core is manufactured, and hence the responsibility for it lies with the core user. This observation leads to the conclusion that in core-based design, testing of the interconnect between cores is important, but testing the internals of the core is at least as important.
- In general, IC pins are scarce. Hence, the number of dedicated test pins should be as low as possible. In the definition of the Boundary Scan Test standard, much effort has been made to reduce the number of pins needed to accommodate the standard. The extra four or five pins form the reason why still many ICs are not equipped with Boundary Scan Test. However, pins are not scarce for embedded cores. As virtual components only have *virtual pins*, extra terminals at the core are more or less for free. This implies that a test access mechanism for embedded cores is hardly limited by the number of additional terminals it might create at the core boundary.

The differences between board test and core test as described above make that, contrary to what some people claim [13, 14], the Boundary Scan Test standard as such is not suited for a standard in the core test domain. The limited test data bandwidth as provided by Boundary Scan Test would lead to excessive long test times for many cores.

This does not mean that Boundary Scan Test is completely useless in defining a test access mechanism for core test. IEEE 1149.1 is the only, successful standard in the test domain and provides an example for the ongoing IEEE standardization activity on core test (IEEE P1500) [4], both in the technical sense, as well as in the standardization organization and political sense. Moreover, many companies have adopted JTAG's TAP and TAP controller to implement all kinds of private test, debug, and emulation modes on their ICs. If these ICs become cores in even larger IC designs, a core test strategy should be capable of handling their hardware-embedded test modes.

5 TestShell

We propose to wrap every IP module in a so-called TESTSHELL. Execution of the pre-computed tests for the embedded IP module only depends on the TESTSHELL and its proper external connections, and does not rely on the contents of other IP modules on the IC. The TESTSHELLs of multiple IP modules also provide the infrastructure for testing the interconnect wiring and logic in between the IP modules.

5.1 Three layers of hierarchy

The TESTSHELL introduces three layers of hierarchy to every core-based IC design (see Figure 3): (1) the IP module, (2) the TESTSHELL, and (3) the host. In the sequel of this paper we use the term *core* to denote the IP module plus TESTSHELL.

The *IP module* is the actual reusable IC design module. The core provider is responsible for its design, as well as its test development and associated design-for-testability (DfT). In our proposal, we do not assume any particular test method or DfT technique to be used for the IP module. Therefore, our proposal is capable of handling legacy IP modules which do not contain any kind of DfT and come with function-based

tests, as well as IP modules for which either ad-hoc or structured DfT techniques, such as scan design and/or built-in self test, have been used.

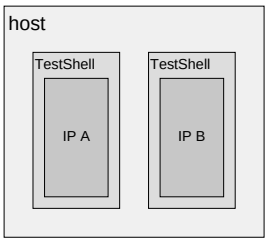

Figure 3: Three hierarchy layers: IP module, TESTSHELL, and host.

The *host* is the environment in which the core is embedded. The host can be either a complete IC, or a design module, meant to become a reusable IP module itself. The host design is the responsibility of the core user. It mainly consists of connecting up the various cores and possibly adding some host-specific logic. The core user is also responsible for the test development and associated DfT for the host. If all cores have a TESTSHELL, the host-level DfT consists of connecting the TESTSHELLs of all cores in a standard way. Test development consists of expanding all IP-level tests to host level, adding interconnect tests, and, possibly, host-level test scheduling to minimize test time. The TESTSHELL is constructed such that test expansion becomes a standard and straightforward transformation.

The TESTSHELL itself is a small interface layer between IP module and host. Its purpose is to facilitate the test expansion task for the core user by creating a standardized interface. The design of the TESTSHELL is influenced by the core provider and his IP design as well as by the core user and his host design. We describe the TESTSHELL in more detail in the next section.

5.2 Host-TestShell interface

The interface between host and TESTSHELL consists of three types of input/output terminals (see Figure 4).

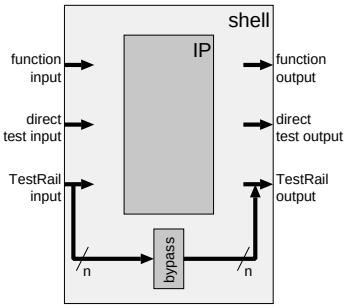

Figure 4: Host-TESTSHELL interface.

- *Function* inputs/outputs.
  The function inputs/outputs of the TESTSHELL correspond one-to-one to the normal inputs and outputs of the IP module. Therefore their number and type are fully determined by the IP module.

- TESTRAIL input/outputs.
  The TESTRAIL is the preferred test access mechanism of the TESTSHELL. The TESTRAIL is capable of handling the test data transport for all synchronous digital tests, including function tests, scan tests, built-in self tests, memory tests, etc. The TESTRAIL has a variable width and it has a bypass mode. The TESTRAIL is described in detail in Section 6.

- *Direct* test inputs/outputs.
  Only test signals which cannot be provided via the TESTRAIL, because they are not synchronous or not digital, come via so-called direct test inputs and outputs. These signals include clocks, asynchronous signals, and analog signals.

5.3 TestShell modes

The TESTSHELL has four mandatory modes.

1. *Function* mode.
   In the Function mode the TESTSHELL is transparent. This mode is meant for normal operation, i.e., when the IC is not tested.

2. *IP Test* mode.
   In the IP Test mode the IP module which is surrounded by this TESTSHELL is being tested. The TESTSHELL takes care that test stimuli are fed from the host inputs to the IP module and vice versa that test responses are transported from the IP module to the host outputs.

3. *Interconnect Test* mode.
   In the Interconnect Test mode, the TESTSHELL is configured such that test stimuli for interconnect logic behind the outputs of this IP module are transported from the host level, while test responses coming from interconnect logic in front of the inputs of this IP module are captured and transported to the host level.

4. *Bypass* mode.
   In the Bypass mode test stimuli and/or responses for other cores are transported through the TESTSHELL, regardless whether or not the IP module has internal transparent modes.

Note that the above modes pertain to the TESTSHELL operation only and are independent of any (test) mode the IP module itself might have. If required, the four mandatory modes of the TESTSHELL can be extended with IP module specific, private (test) modes. In Section 7 we describe how the various modes of the TESTSHELL are controlled.

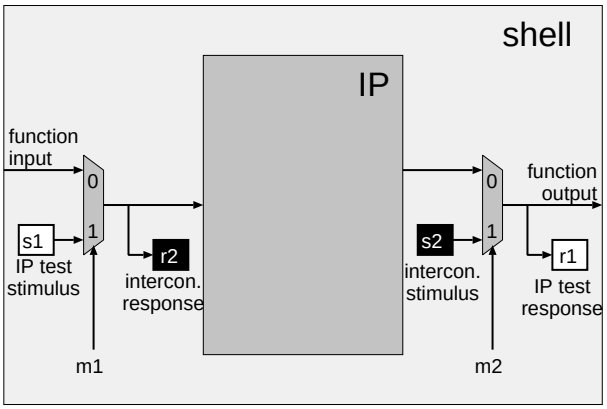

Figure 5: Conceptual view of the TESTSHELL (bypass not shown).

Figure 5 gives a conceptual view of the TESTSHELL. In Function mode, the multiplexers $m1$ and $m2$ are set to `00`. In this way, the TESTSHELL becomes transparent; function inputs and outputs are logically directly connected to the IP module. The propagation delay of the multiplexers in the normal access path may affect the performance, and hence should be taken into account. In IP Test mode, the multiplexers are set to `10`. Test stimuli for the IP module come from $s1$, which represents a connection to either the TESTRAIL or a direct test input, depending on the signal type. Test responses of the IP module are captured in $r1$, which likewise represents either TESTRAIL or direct test output. In Interconnect Test mode, the multiplexers are set to `01`. $s2$ serves to apply test stimuli to the TESTSHELL output. $s2$ represents a connection to the TESTRAIL. Likewise, $r2$ represents a connection to the TESTRAIL which is used to capture the test response coming via the TESTSHELL input.

Note that the above is a *conceptual* description of the TESTSHELL only. It is meant to describe the functional behavior of the TESTSHELL, and does not relate to its actual implementation. The actual implementation leaves room for optimization of silicon area costs and impact of performance. Also, it should contain provisions for multiple clock domains and clock skew within a core and between cores. These details are not discussed in this paper.

6 TestRail

The TESTRAIL is the preferred test data transport mechanism of our approach. It is capable of transporting test stimuli and responses for synchronous digital tests. Every TESTSHELL has a TESTRAIL of n wires wide ($n \geq 0$). However, at host-level, the TESTRAILs of the various cores can be connected in many different ways, and therefore provide the host designer the flexibility to make trade-offs between test time and silicon area.

6.1 TestRail width

The width of a TESTRAIL for a particular core has to be the outcome of negotiations between the core provider and the core user. The first brings IP-level considerations into the discussion, while the second is aware of any host-level considerations. The TESTRAIL widths of all cores of a host design relate to the following cost factors.

- *Host pins*.
  The number of available host pins is an important limiting factor w.r.t. the maximum TESTRAIL width. Also the direction type of the pins, i.e., input, output, or bidirectional, has to be taken into account, as every TESTRAIL wire needs one input and one output pin.

- *Test time*.
  The width of a TESTRAIL determines its maximum test data bandwidth. For given tests, the width of the TESTRAIL therefore determines the number of host-level test vectors and hence the test application time, as well as the required size of the pin memories of the test equipment. Host-level TESTRAIL configurations determine the possibilities for testing multiple cores in parallel and hence influence the outcome of test scheduling.

- *Silicon area*.
  Wiring of the TESTRAIL occupies silicon area. The width of the individual TESTRAILs and the host-level configuration of multiple TESTRAILs influence the area needed for this wiring.

In general, the number of host pins available to accommodate test signals is given. Within that constraint, a trade-off between test time and silicon area can be made. If the precomputed test of a certain IP module consists of only a small number of bits, it needs only a narrow TESTRAIL. This is for example the case if an IP module has a built-in self test. It is better to devote the scarce host pins and silicon area to other IP modules. In Aerts & Marinissen [15], the test time of various scan chain architectures, which can all be seen as

implementation examples of the general TESTRAIL concept, are analyzed.

6.2 Host-level TestRail connection

The TESTRAIL concept is very flexible, in the sense that it allows many configurations at host level. One end of the spectrum of possible TESTRAIL connections is the case in which every core has its private TESTRAIL and all TESTRAILs are multiplexed onto the available host pins; this corresponds to the method described in [8]. The other end of the spectrum is formed by the case in which all TESTRAILs of the individual cores are concatenated into one host-level TESTRAIL.

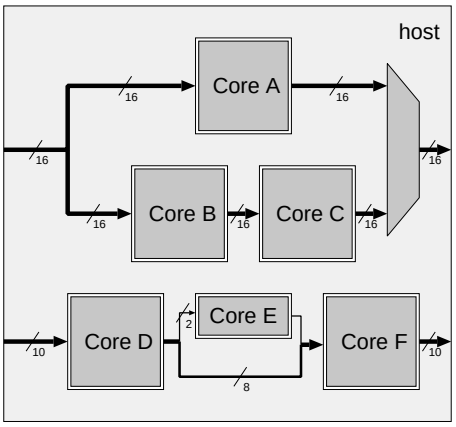

Figure 6: Example of possible host-level TESTRAIL connections.

Figure 6 depicts several ways in which TESTRAILs of individual cores can be connected at host level. The TESTRAILs in the example host have various widths. Core A has a private TESTRAIL of width 16, while the TESTRAILs of cores B and C, both also of width 16, are concatenated. TESTRAILs can fork out; the TESTRAIL of core D of width 10 forks out to the TESTRAIL of core E of width 2 and another 8 lines. TESTRAILs can also merge together; the TESTRAIL of core E and the 8 lines coming from core D merge together in the TESTRAIL of core F of width 10. A TESTRAIL can have its private pins, such as the TESTRAIL through cores D, E, and F; alternatively, the inputs and outputs of multiple TESTRAILs can be multiplexed, such as is the case with the TESTRAIL through core A and the TESTRAIL through cores B and C. On top of this, TESTRAIL inputs and outputs can be combined with normal function pins, a practice well-known from ordinary scan design. N.B. Note that Figure 6 is only meant to illustrate the flexibility of the TESTRAIL concept; many real ICs will have more straightforward TESTRAIL implementations!

6.3 IP-level TestRail connection

Within the TESTSHELL, a TESTRAIL of given width is connected to the input and output terminals of the IP module. Figure 7 shows the three basic forms of connection: (1) parallel, (2) serial, and (3) compressed.

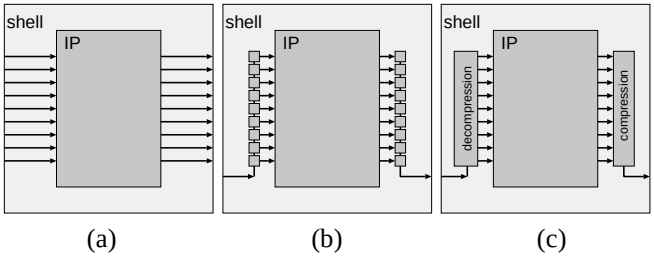

Figure 7: IP-level TESTRAIL connections: (a) parallel, (b) serial, and (c) compressed.

Parallel connection means that the TESTRAIL is one-to-one connected to the terminals of the IP module. In a *serial* connection one TESTRAIL wire is connected to multiple IP terminals by means of a shift register which serves as serial-parallel or parallel-serial converter. A *compressed* connection refers to decompression hardware at IP inputs or compression hardware at IP outputs. Note that decompression at IP inputs is in practice only possible for very regular sequences of stimuli, such as is the case for memories. Compression of IP outputs, e.g., through a Multiple-Input Shift Register (MISR) or Exclusive-OR tree, is always possible. Combinations of the three types of connections are also possible; e.g., inputs partly parallel, partly serial, while outputs are compressed.

Which type of connection should be implemented for a particular IP module, depends on the width of the available TESTRAIL and the number of input and outputs terminals of the IP module in question. If the TESTRAIL width is sufficient, all IP terminals can get a parallel connection to the TESTRAIL. If the TESTRAIL width is not sufficient for this, the best strategy w.r.t. test time reduction is to give a parallel connection to those inputs and outputs that are used to transport most test data, while others can do with a serial connection.

6.4 TestRail bypass

By default, the TESTRAIL has a bypass mode. The bypass is useful if multiple cores are connected serially into one TESTRAIL. The cores in this TESTRAIL which are not tested, can be put into bypass mode to create or shorten an access path to the core-under-test. An arbitrary number of cores can be serially connected into one TESTRAIL. In order to prevent too long propagation delays, we have implemented the bypass cell of Figure 4 as a register, clocked

by the TESTSHELL clock TCK. From this implementation evolves the limitation that the TESTRAIL is only capable of transporting synchronous digital signals.

There are situations in which the bypass mode is not required at all, or in which the bypass cell can be implemented differently. For example, a core with a private TESTRAIL, i.e., no other cores are connected into that same TESTRAIL, does not need a bypass mode. In that case the TESTRAIL and direct test access degenerate into the same thing, implying that such a TESTRAIL would indeed be capable of transporting not only synchronous digital test signals, but asynchronous signals and analog signals as well.

7 Test Control Mechanism

The TESTSHELL contains a standardized test control mechanism (TCM). This test control mechanism is primarily meant to control the operation of the TESTSHELL. However, it can be extended to control the operation of the IP module embedded within the TESTSHELL as well. This section describes the conceptual implementation of the TCM.

Two types of test control signals exist.

1. *Pseudo-static* test control signals.
   These signals set up the conditions for a certain test, and remain stable during that entire test.
2. *Dynamic* test control signals.
   These signals change value even during the execution of one test pattern. An example of such a signal is the scan enable signal in the case of scan design.

Setting up a new set of pseudo-static test control signals may take multiple clock cycles. Dynamic test control signals, on the other hand, should be able to change value from one clock cycle to the next. For example, switching internal scan chains from shift mode to normal mode or vice versa may not take several clock cycles. The TCM described in this paper is meant for pseudo-static test control signals. The dynamic test control signals can be treated as ordinary test data signals.

Every TESTSHELL only has a few pseudo-static test control signals. Fast access is not required, because these signals change values only very infrequently. In order to reduce the burden of the TCM on the available host pins, we propose to apply the pseudo-static test control signals via a serial access mechanism. Every TESTSHELL contains a register pair, consisting of a *shift* and an *update* register, similar to the one used in the Boundary Scan Test (1149.1) standard [10]. Both registers contain a number of bits equal to the required number of pseudo-static test control signals. We can consider the mechanism to consist of *bit slices* (see Figure 8).

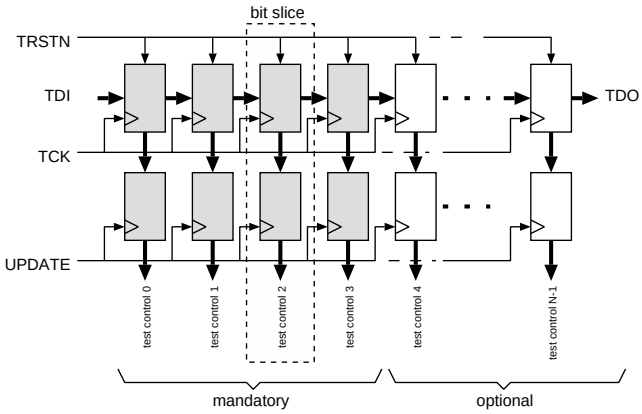

Figure 8: Bit-slice based test control mechanism.

The TCM contained in the TESTSHELL has four mandatory bit slices, corresponding to the four modes of the TESTSHELL as described in Section 5: Function, IP Test, Interconnect Test, and Bypass. In Figure 8, these four bit slices are shaded gray. If required, these four mandatory bit slices can be extended with other bit slices, representing internal test modes of the IP module.

Every core has a standard test control interface, consisting of the following terminals: TDI, TDO, TCK, TRSTN, and UPDATE.

Figure 9 depicts two cores with their respective TCMs. The TCM of IP module A contains six bit slices; apart from the four mandatory bit slices, two other bit slices control test modes internal to A. IP module B contains an internal TCM, and hence the TCM at TESTSHELL-level only has the four mandatory bits that control the TESTSHELL operation.

At host level, the core user is free to connect the TCMs of the various cores as he wants. One possibility is to daisy-chain the TCMs into one long shift/update register. This is done by connecting the TDO output of one core to the TDI input of another, as depicted in Figure 9. In order to increase the diagnostic resolution in case of failing silicon, the core user could choose to provide a bypass for all core-level TCMs.

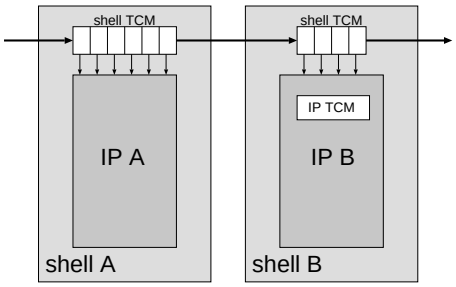

Figure 9: Example with two daisy-chained core.

8 Application Example

In this section we give an application example of the TESTSHELL with TESTRAIL. In this example we consider an IC which consists of three cores, named A, B, and C. In order to test this IC, we need four tests.

1. Test for IP A.
   IP A has scan design as its embedded DfT methodology. IP A has nine inputs, nine outputs, and four internal scan chains.

2. Test for IP B.
   IP B has Built-In Self Test (BIST) as its embedded DfT methodology. B's BIST has to be initialized, and will then execute autonomously, after which a signature has to be captured.

3. Test for IP C.
   IP C is tested through a (synchronous) function test, in which all inputs and outputs of C are involved. C has nine inputs and nine outputs.

4. Global interconnect test for the IC.

All three IP modules are packaged with a TESTSHELL and a TESTRAIL of width five. The core user has decided to make one IC-level TESTRAIL, in which the TESTRAILs of the three cores are concatenated.

For all four tests, the TESTRAIL is used to transport test stimuli and responses. This is possible, because all four tests are synchronous digital tests. For Tests 1-3, the TESTSHELL of the core under test (either A, B, or C) is put into IP Test mode, while the other two TESTSHELLs are put into Bypass mode (see Figure 10a). For Test 4, the TESTSHELLs of all cores are put into Interconnect Test mode (see Figure 10b).

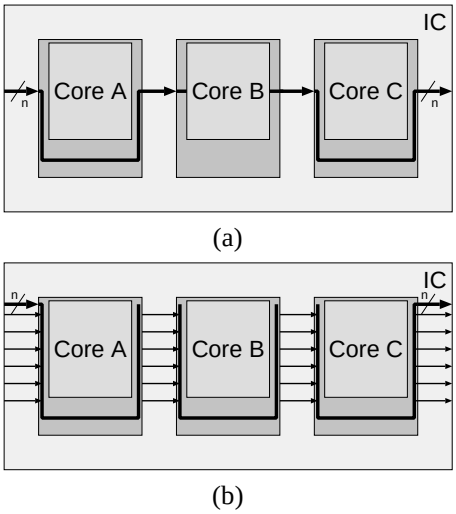

Figure 10: Examples of (a) test of IP module B, and (b) global interconnect test.

Figure 11 shows the TESTRAIL connections, both within the respective TESTSHELLs, as well as at IC level. Cores A and C are connected to the full TESTRAIL of width five. The BISTed core B is connected to only one of the TESTRAIL wires; the other four bypass B outside its TESTSHELL.

The four scan inputs (outputs) of IP module A are used to transport much more test data than the five other inputs (outputs). Hence, we provide the scan inputs (outputs) with a parallel connection to the TESTRAIL, while the other inputs (outputs) are serially connected to the TESTRAIL.

IP module B is tested by BIST. Hence, only very little test data has to be transported to and from B. Therefore, core B is only connected to one of the five TESTRAIL wires.

IP module C is tested functionally. All nine inputs (outputs) transport an equal amount of test data, as they are all every clock cycle of the test involved. However, the TESTRAIL has only width five. Hence, four of the five inputs (outputs) require a serial TESTRAIL connection, simply because there are not enough wires to connect all inputs in a parallel fashion. If the function test of IP C requires that all test patterns are applied subsequently, we have to add clock holding circuitry, such that at every IP clock cycle, one pattern can be applied.

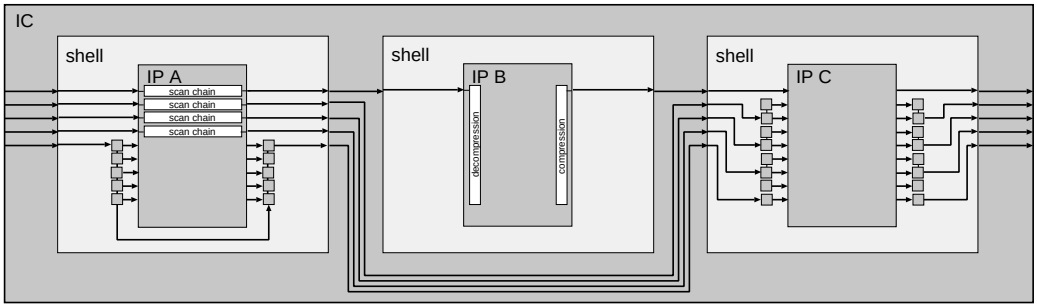

Figure 11: TESTRAIL connections for example IC containing three cores.

9 Conclusion

Reuse of large design modules, termed *cores*, aims at reduced time-to-market for complex systems-on-chips. The test development for such core-based ICs can be speeded up by the same reuse paradigm. This implies that a core provider should, together with the core's design description, deliver a pre-computed set of tests for the core. The core user is responsible for providing on-chip hardware such that the test patterns of the pre-computed tests can be applied to the embedded core via the IC pins.

In this paper we presented the concept of a structured and scalable access mechanism for embedded cores. We propose to wrap every reusable IP module into a TESTSHELL, a thin interface layer between IP module and host environment. The TESTSHELL has various modes of operation: Function, IP Test, Interconnect Test, and Bypass.

The main source of test data access of the TESTSHELL is formed by the TESTRAIL. The TESTRAIL is capable of handling all digital synchronous test signals from IC pins to core-under test and vice versa. The TESTRAIL is structured in the sense that it has a standard interface and implementation. The TESTRAIL is scalable in the sense that its width can be adapted to the test data volume of the core-under-test. Therefore, the TESTRAIL concept enables trade-offs between the number of IC pins, test time, and silicon area. IP terminals which cannot be connected to the TESTRAIL get *direct* test access, which means a direct connection to IC pins.

The modes of operation of the TESTSHELL are controlled via a dedicated test control mechanism (TCM). The TCM is structured in the sense that it has a standard interface and implementation. The TCM is scalable in the sense that it can be extended to not only control the operation of the TESTSHELL itself, but, if required, also the internal test modes of the IP module.

We feel the test access mechanism as presented in this paper is well-suited to form the conceptual basis for a world-wide standard, such as is pursued by IEEE P1500.

Acknowledgements

The six authors together form the Philips Core Test Action Group (CTAG), responsible for defining Philips-internal core test strategies and tool development. The authors gratefully acknowledge the cooperation and comments of our colleagues Emile Aarts, Keith Baker, Bart De Loore, Paul Merkus, Piet Mourik, and René Segers.

This work was partially supported by the A401 project under the Medea programme.

References

[1] R. Chandramouli and Stephen Pateras. Testing Systems on a Chip. *IEEE Spectrum*, pages 42–47, November 1996.

[2] Marcel Boosten and Harro Jacobs. Test Protocol Expansion: Memory Handling and Efficiency Improvements. Master's thesis, Eindhoven University of Technology, Eindhoven, The Netherlands, June 1994.

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

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

[5] 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.

[6] Erik Jan Marinissen, Krijn Kuiper, and Clemens Wouters. Test Protocol Expansion in Hierarchical Macro Testing. In *Proceedings IEEE European Test Conference*, pages 28–36, April 1993.

[7] 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.

[8] 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.

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

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

[11] Harry Bleeker, Peter van den Eijnden, and Frans de Jong. *Boundary-Scan Test - A Practical Approach*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1993.

[12] Colin Maunder and Rodham E. Tullos. *The Test Access Port and Boundary Scan Architecture*. IEEE Computer Society Press Tutorial. IEEE Computer Society, Los Alamitos, 1990. ISBN 0-8186-9070-4.

[13] 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.

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

[15] 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.