An Industrial Approach to Core-Based System Chip Testing

Erik Jan Marinissen
Philips Research

Abstract: System chips are increasingly being designed by embedding reusable pre-designed and pre-verified cores. Modular, core-based test development is an attractive proposition for such large and complex ICs. This paper outlines the approach developed and used in Philips for core-based testing. It consists of four components: (1) a standardized set of test deliverables for a core, (2) standardized on-chip test access hardware, (3) a tool for translation of core tests into SOC tests, and (4) a tool for test scheduling.

Key words: system chip, manufacturing test, test wrapper, test access mechanism, test expansion, test scheduling, Macro Test

1. INTRODUCTION

Modern semiconductor process technologies enable the manufacturing of a complete system on one single die, the so-called *system chip* or SOC. Such system chips typically are very large ICs, consisting of millions of transistors, and containing a variety of hardware modules. In order to design these large and complex system chips in a timely manner and leverage from external design expertise, increasingly reusable cores are utilized. *Cores* are pre-designed and pre-verified design modules, meant to be reused in multiple SOC designs. Examples of cores are CPUs, DSPs, media co-processors, communication modules, memories, and mixed-signal modules. Core-based design divides the IC design community into *core providers* and *core users*. Core-based design takes place within companies, as well as between companies, i.e., one company acts as core provider, while another

---

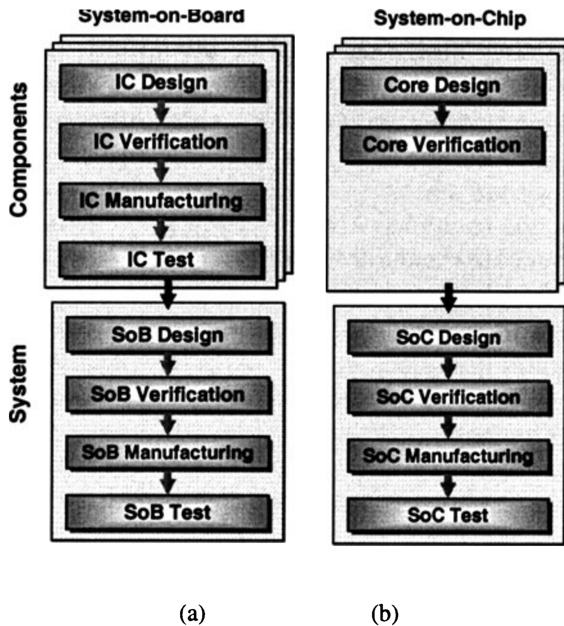
The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35597-9_40

M. Robert et al. (eds.), *SOC Design Methodologies*
© IFIP International Federation for Information Processing 2002

company is the core user. Especially in the latter scenario, legal issues with respect to intellectual property rights and liability come to the table.

Due to imperfections in their manufacturing process, all electronic systems need to be tested for manufacturing defects. System chips are no exception to that rule. In traditional system *boards*, the components (ICs) used are tested by their provider (the IC manufacturer), and hence the system integrator only needs to test the rest of the board — often mainly interconnect wiring (cf. Figure 1(a)). For system *chips*, this situation is quite different. The components (cores) cannot be tested for manufacturing defects by their providers, as what is transferred from provider to user is merely a description, and hence not yet manufactured [1]. Manufacturing only takes place once the entire system chip is assembled (cf. Figure 1(b)). Therefore, a system chip integrator is responsible for testing both the user-defined circuitry surrounding the embedded cores, but also the embedded cores themselves. Even though the core provider might have tested his product for design errors and other users of the same core might have tested it in their ICs for manufacturing defects, that does not take away the obligation of the core user to test his system chip for manufacturing defects.

Testing of core-based system chips is best done in a core-based fashion: the core providers prepare their products with the right design-for-testability hardware and create test patterns for the cores, while the system chip integrator adds SOC-level design-for-testability and creates a system chip test using the core-level tests as building blocks [1]. In this core-based way of SOC test development, the system chip integrator does not need to be bothered with the content of the cores he is using in order to develop a test of sufficient quality. In many cases, core providers also do not want to share the implementation details of their cores in order to protect their intellectual property in the design of the core. They might deliver their cores as layouts or encrypted netlists, without providing RTL or netlist-level sources. In such a situation, the core user needs to rely on the tests as provided by the core provider, as he does not have access to the core's implementation details and hence cannot generate the tests himself. Furthermore, core-based testing allows for test reuse. A test developed once for a given core, can be reused for many instantiations of that core in different system chips, hence optimizing the usage of scarce engineering resources. And, if proper procedures and tools are in place, this way of working should also save precious development time on the side of the system chip integrator, thereby contributing to a shorter time-to-market.



(a) (b)

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

Efficient design of system chips is very important to Philips Semiconductors. Hence, Philips has strongly embraced the paradigms of core-based system chip design, and, consequently, core-based testing. This paper details what solutions have been put into place within Philips to enable effective and efficient core-based testing.

The sequel of this paper is organized as follows. Section 2 lists the three main challenges involved in core-based testing and their proposed solutions. Subsequently, Section 3 provides the overview of and motivation for the Philips-internal solutions to these challenges. Sections 4, 5, and 6 provide more detail on each of the solutions. Section 7 briefly describes some experiences with using this core-based test approach on industrial SOCs. Finally, Section 8 concludes this paper.

2. CHALLENGES IN CORE-BASED TESTING

The traditional challenge for IC test engineers is to find a right balance between (1) test quality, (2) test costs, and (3) test development time.

Next to this traditional challenge, which of course remains in full force, also for core-based SOC designs, there are new challenges, specifically related to core-based testing. Marinissen and Zorian [2] listed the following core-based test challenges.

1. **Distributed test development.**
   Core-based design and test development is an effort distributed over multiple parties, spread over location, time, and company. The joint responsibility of all parties involved for the SOC test requires the transfer of 'core test knowledge' from core provider to core user.
   Required solution: A standardized set of deliverables, preferably delivered in standardized directory structures and file formats.
2. **Test access to embedded cores.**
   Cores are typically deeply embedded inside the SOC design. Testing takes place at the SOC pins, and hence electrical test access has to be created from the SOC pins to the core-under-test and vice versa.
   Zorian et al. [1] introduced a generic conceptual architecture for test access to embedded cores. It consists of (1) a *source* and *sink*, that generate the test stimuli and evaluate the test responses, (2) a *test access mechanism* (TAM) that bridges the physical distance between source/sink and core, and (3) a core test *wrapper*, that isolates the core during testing and provides switching between functional and test access.
   Required solution:
   - A standardized on-chip test access infrastructure.
   - Tools for test translation from core to SOC.
3. **SOC-Level test optimisations.**
   A modular SOC test approach allows optimization trade-offs. Which cores will be tested as stand-alone units, and which cores will be merged and tested as part of the overall SOC-level test? How to design and optimise the test access infrastructure? How to schedule the various tests, such that the test application time is minimized, but for example a power dissipation budget is not exceeded?
   Required solution: Tools to evaluate and implement the various trade-offs.

3. OVERVIEW OF PHILIPS' CORE-BASED TEST APPROACH

Modular testing, in which modules are tested as separate entities, is nothing new in Philips. Already in the mid 1980s, *Macro Test* was defined by Beenker et al. [3]. Already back then, ICs were composed of multiple modules, termed *macros*. These macros had different circuit structures, such

as logic, memories, PLAs, register files, etc. Different circuit structures exhibit different defect behavior, and hence require dedicated test pattern generation and independent testing. Tools were developed for automatic test expansion from macro level to IC level, and those have been successfully used for numerous industrial ICs [4].

In finding test access paths from the embedded macro terminals to the IC pins, one inevitably has to make use of other on-chip circuitry. In 1980-90, adding dedicated additional test access hardware was considered too expensive. Hence, the tool for test protocol expansion was made such that it was capable of utilizing existing functional access paths through neighboring macros. It would make use of the scan chains in neigboring macros, as well as exploit functional transparent paths, both at the gate level as well as at higher levels of abstraction.

Times have changed since the mid 1980s. What used to be entire ICs are now only cores on a system chip. The sheer size of some system chips is impractical or intractable for some DfT insertion and/or test generation tools, and hence this asks for a 'divide-and-conquer' solution. Some cores come from external sources and have their implementation details hidden, i.e., we do not have available transparent paths through them for test access to neighboring cores. Silicon area is still a cost factor, but has relatively decreased in importance, whereas time-to-market has gained importance.

In 1997, Philips formed an internal team to study core-based testing. This team was named CTAG: Core Test Action Group. Its mission was to define additional methods and tools, on top of Macro Test, that would improve and ease core-based testing. CTAG defined two items.

- *Standardized deliverables.* These standardized deliverables have been integrated into the CoReUse Standards & Constraints, that define similar items in other areas than testing.
- *Standardized wrapper and TAM*, named TestShell and TestRail. These have been documented in a Philips-internal standard. Some aspects of TestShell and TestRail have been patented. Furthermore, tools have been developed for the automatic generation and verification of TestShells and TestRails.

TestShell and TestRail are dedicated on-chip test access hardware, and therefore mark a clear difference with the early years of Macro Test usage, when such dedicated access hardware was considered to be too expensive. TestShell and TestRail ensure guaranteed test access; the accessibility of a core does not depend on neighboring circuitry, but is handled via dedicated 'highways'. This also makes the task of test protocol expansion a lot easier and hence contributes to reduced test development times.

Table 1 summarizes the core-based test challenges and required solutions, as discussed in Section 2 and adds to that an overview of the Philips-internal variants of those solutions.

Table 1. Summary of challenges and solutions in core-based testing.

	Challenge	Required Solution	Philips Solution
1.	Distributed test development	Standardized set of deliverables	CoReUse Standards & Constraints
2.	Test access to embedded cores	Standardized on-chip access hardware +	CTAG TestShell and TestRail
		Tools for test translation	CAT Test Protocol Expansion
3.	SOC-level optimization	Tools for evaluating trade-offs	CAT Test Protocol Scheduling

4. COREUSE STANDARDS AND CONSTRAINTS

CoReUse *Standards* define required and optional delivery views for soft, firm, and hard cores, as well as standardized directory structures and file naming conventions. The standard delivery views ensure that core providers know what should be delivered, and core users know what they can expect. Delivery views include the following.

- Inserted DfT, such as scan chains or BIST.
- Inserted TestShell and its verification report.
- Test pattern lists and their respective fault coverages.
- Test protocols for the various test pattern lists.
- Diagnostic information.

CoReUse *Constraints* define requirements regarding the contents of the delivered files. There are Constraints with respect to RTL design, Verilog and VHDL, Verilog coding, VHDL coding, coding of scripts, test bench design and test suite development, packaging, test and testability, and testability port naming conventions.

5. CTAG TESTSHELL AND TESTRAIL

The CTAG TestShell, as depicted in Figure 2, has been published first by Marinissen et al. [5]. The TestShell supports four basic modes: (1) normal functional mode, (2) an inward-facing IP test mode, (3) and outward-facing interconnect test mode, and (4) the bypass mode.

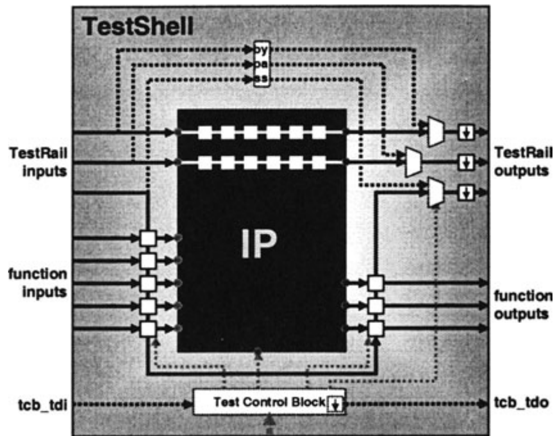


Figure 2. Conceptual view of the TestShell.

The TestShell has the following interface.

- Functional inputs and outputs, directly corresponding to the functional inputs and outputs of the unwrapped core.
- A one-bit test control input and output.
- A scalable multi-bit test data input and output, also referred to as the TestRail plug.

The TestShell consists of the following components.

- A *Test Cell* for every core terminal. The test cell provides controllability as well as observability.
- An (optional) *Bypass Register*, which allows a TAM to bypass core and wrapper, in order to test another core that is connected to the same TAM.
- A *Test Control Block* (TCB). The TCB has a bit-slice nature and consists of a shift and an update register. The TCB is primarily meant to control the operation of the TestShell, through several mandatory bit slices. Additional user-defined bit slices can be added for control of core-internal test modes.

A TestRail is a dedicated TAM in between TestShells. In principle, a TestShell is connected to the same TestRail at both input and output. Therefore, the TAM input plug and the TAM output plug of a TestShell normally have the same width. At SOC-level, the number, widths, and configuration of the TestRail is completely user-defined. In [6], three alternative TAM configurations were published (see Figure 3); TestRail supports all of these and others.

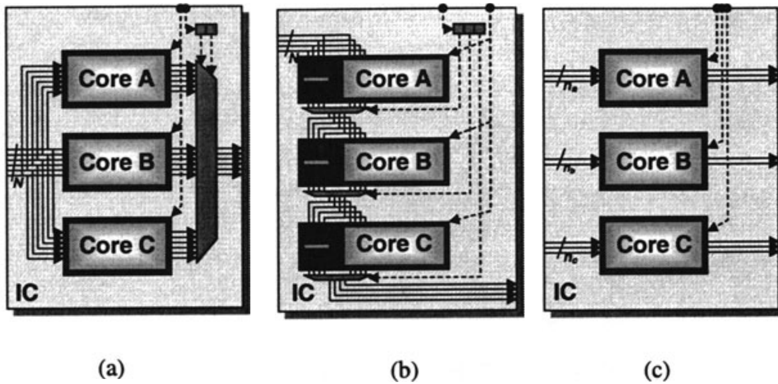


Figure 3. Examples of three TAM configurations [6]: (a) Multiplexing, (b) Daisychain, and (c) Distribution architectures.

6. MACRO TEST TOOLS

Macro Test has originally been defined by Beenker et al. [3]. The application of its concepts in the domain of core-based testing have been reported in [7,8].

One of the fundamental concepts behind Macro Test is that a test of a core is split up into a *test protocol* and a list of *test patterns*. The test protocol describes how to apply one test pattern in a pattern-value independent way. It does not specify how many patterns need to be applied, nor what the data content of the test patterns is.

Initially, both test protocol and test patterns are defined at the core terminals, as if the core terminals are directly accessible. While the bulk of the data, contained in the test patterns, remains untouched, subsequent tasks that translate the core-level test into a SOC-level test are performed on test protocols only.

Test protocol expansion translates the initial test protocol, defined at the core terminals, into a SOC-level test protocol, defined at the IC pins. This process involves exhaustive path searching. It is capable of exploiting *transfer* descriptions of neighboring cores at any level of hierarchical abstraction. The general test protocol expansion problem has been proven to be *NP*-hard. However, the test protocol expansion tool has a built-in preference for test access via TestShells and TestRails, as they provide a dedicated test access infrastructure and hence guarantee an efficient and successful expansion.

Test protocol scheduling tries to overlap the expanded test protocols of various cores in order to minimize the resulting test vector set and test

application time [8]. Our definition of the test protocol scheduling problem has been shown to be *NP*-hard and similar to the No-Wait Job Shop Scheduling problem [9].

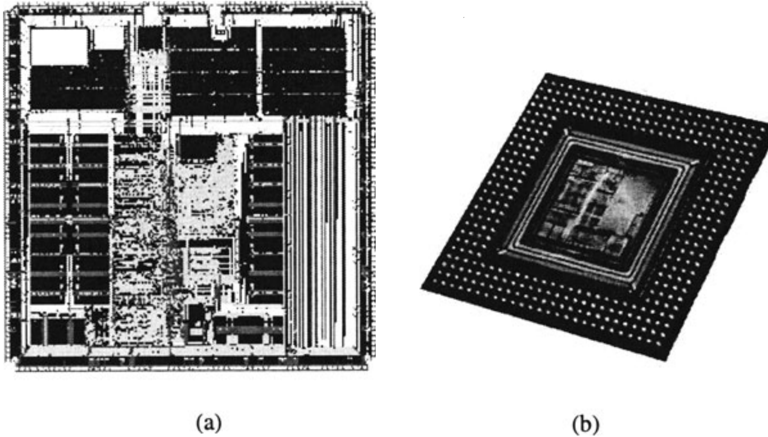
Test assembly creates test vectors by filling in the test patterns into the corresponding test protocols. Our test assembly tool can write out test vectors for netlist simulation (together with a simulation test bench), or for the various ATE models in use. Test assembly can be performed on the unexpanded test protocols, in order to create verification suites for the core's test patterns. Also, test assembly can be performed for the expanded and scheduled test protocols, to generate the final test vector set for a core at SOC level [7,8].

7. APPLICATION EXAMPLES

The core-based test approach as described in this paper has been applied to many industrial Philips SOCs. In this section, we briefly report on two case studies.

The first case study reported by Arendsen and Lousberg [10] applied a core-based test strategy to a Digital Still Camera (DSC) IC. The IC was manufactured in a 0.35 μm five metal layer CMOS technology with a total die area of 50 mm^2 (see Figure 4(a)). It contained a mix of nine hard, firm, and soft cores. One of the cores, a bus control unit, was considered too small to be treated as a stand-alone unit, and hence tested as part of the overall interconnect test. The other eight cores were tested as individual cores. TestShells and distributed TestRails were added to them, and their SOC-level tests were generated by means of the Macro Test tools.

The design benefits included a clear separation of tasks between core provider and core user, easy integration with respect to testing, and strongly reduced ATPG run times due to the divide-and-conquer strategy. The relative area costs are listed in Table 2. The table shows that the additional area costs for TestShell and TestRail depend strongly on the size of the core. In this case, the additional area costs attributed to TestShell and TestRail varied between 0.5% for the largest core and 22.1% for the smallest core. In total 7.7% of the total chip area was devoted to test infrastructure; 4.5% for the core-internal scan chains and an additional 3.2% for TestShell and TestRail, which enabled the core-based test approach. These additional area costs were considered acceptable in the view of the associated benefits.



(a) (b)

Figure 4. Layout of DSC-IC (a) and chip photo of CPA-IC (b).

Table 2. Relative DfT area costs for DSC-IC

Core	Core Area as Part of Total IC	Functional Area	DfT Area		
			Core Internal	TestShell + TestRail	Total
RISC Core 1	69.7%	96.8%	2.7%	0.5%	3.2%
RISC Core 2	11.2%	92.3%	5.7%	2.0%	7.7%
Peripheral 2	7.5%	76.7%	11.6%	11.7%	23.3%
UART (2×)	4.9%	77.6%	9.6%	12.7%	22.3%
SDRAM Interface	3.0%	83.0%	12.0%	5.0%	17.0%
Debug Support Unit	1.8%	72.8%	10.8%	16.4%	27.2%
Peripheral 1	1.5%	69.7%	8.1%	22.1%	30.2%
Bus Control Unit	0.4%	90.3%	9.7%	0.0%	9.7%
Total	100.0%	92.3%	4.5%	3.2%	7.7%

A second case study was reported by Van Beers and Van Herten [11]. It involves a Co-Processor Array (CPA) for digital video applications, consisting of twelve large video co-processors and a switch matrix, and manufactured as 7.6 M transistors (incl. 0.5 M logic gates and 90 K flip flops) in a 0.35 μm five metal layer CMOS technology with a total die area of 170 mm^2 (see Figure 4(b)). Also this IC was tested in a core-based fashion. In total 13 cores were equiped with TestShells and connected into one 48-bit wide daisychained TestRail.

8. CONCLUSION

In order to improve on design productivity and import external expertise, large system chips are increasingly being designed by embedding reusable

pre-designed and pre-verified cores. For such core-based system chips, modular, core-based, test development is an attractive proposition.

Philips' core-based test strategy includes these four solutions.

1. The CoReUse Standards and Constraints define standardized deliverables, both with respect to form and content.
2. The CTAG TestShell and TestRail are the Philips-internally standardized, but scalable wrapper and TAM.
3. Test protocol expansion is the Philips tool for test translation. By working only with test protocols and abstracting from the actual test pattern values, the computational complexity involved is significantly reduced.
4. Test protocol scheduling is the Philips tool that implements one of the possible tools for SOC-level optimization.

Philips' core-based test strategy has been and continues to be successfully used on numerous industrial SOC designs, and has become the de-facto DfT and test generation strategy within the company. For the future we expect that some of our internal standards will be replaced by industry-wide standards, such as, IEEE P1500 SECT [12].

9. ACKNOWLEDGEMENTS

I gratefully acknowledge the contributions of many Philips colleagues to the achievements described in this paper. Special thanks go to my original fellow members of CTAG: Robert Arendsen, Gerard Bos, Hans Dingemanse, Maurice Lousberg, and Clemens Wouters. The data on the various applications of core-based testing have been obtained by Robert Arendsen, Jos van Beers, and Harry van Herten.

10. REFERENCES

[1] Y. Zorian, EJ. Marinissen, and Sujit Dey. Testing Embedded-Core Based System Chips. In Proceedings IEEE International Test Conference, pages 130–143, Washington, DC, October 1998.

[2] E.J. Marinissen and Y. Zorian. Challenges in Testing Core-Based System ICs. IEEE Communications Magazine, 37(6):104–109, June 1999.

[3] F. Beenker, K. van Eerdewijk, R. Gerritsen, F. Peacock, and M. van der Star. Macro Testing: Unifying IC and Board Test. IEEE Design & Test of Computers, Vol. 3(No. 4):26–32, December 1986.

[4] F. Bouwman, S. Oostdijk, R. Stans, B. Bennetts, and F. Beenker. Macro Testability; The Results of Production Device Applications. In Proceedings IEEE International Test Conference, pages 232–241, September 1992.

[5] E.J. Marinissen et al. A Structured And Scalable Mechanism for Test Access to Embedded Reusable Cores. In Proceedings IEEE International Test Conference, pages 284–293, Washington, DC, October 1998.
[6] J. Aerts and E.J. Marinissen. Scan Chain Design for Test Time Reduction in Core-Based ICs. In Proceedings IEEE International Test Conference, pages 448–457, Washington, DC, October 1998.
[7] E.J. Marinissen and M. Lousberg. The Role of Test Protocols in Testing Embedded-Core-Based System ICs. In Proceedings IEEE European Test Workshop, pages 70–75, Konstanz, Germany, May 1999.
[8] E.J. Marinissen. The Role of Test Protocols in Automated Test Generation for Embedded-Core-Based System ICs. Journal of Electronic Testing: Theory and Applications, Vol. 18(No. 4), August 2002.
[9] M. Pinedo. Scheduling - Theory, Algorithms, and Systems. Prentice Hall, Englewood Cliffs, New Jersey, 1995.
[10] R. Arendsen and M. Lousberg. Core Based Test for a System on Chip Architecture Framework. In Digest of Papers of IEEE International Workshop on Testing Embedded Core-Based Systems, pages 5.1–1–8, Washington, DC, October 1998.
[11] J. van Beers and H. van Herten. Test Features of a Core-Based Co-Processor Array for Video Applications. In Proceedings IEEE International Test Conference, pages 638–647, Atlantic City, NJ, September 1999.
[12] E.J. Marinissen, R. Kapur, and Y. Zorian. On Using IEEE P1500 SECT for Test Plug-n-Play. In Proceedings IEEE International Test Conference, pages 770–777, Atlantic City, NJ, October 2000.