A Non-Intrusive Isolation Approach for Soft Cores

Ozgur Sinanoglu

Math & Computer Science Department
Kuwait University
Safat, Kuwait 13060
ozgur@sci.kuniv.edu.kw

Tsvetomir Petrov

Qualcomm CDMA Technologies
Qualcomm, Inc.
San Diego, CA
tpetrov@qualcomm.com

ABSTRACT

Cost effective SOC test strongly hinges on parallel, independent test of SOC cores, which can only be ensured through proper core isolation techniques. While a core isolation mechanism can provide controllability and observability at the core I/O interface, its implementation may have various implications on area, functional timing, test time and data volume, and at-speed coverage on the core interface. In this paper, we propose a non-intrusive core isolation technique that is based on the utilization of existing core registers for isolating the core. We provide a core register partitioning algorithm that is capable of identifying the core interface registers, and of robustly isolating a core, resulting in a computationally efficient core isolation implementation that is area and performance efficient at the same time. The proposed isolation technique also ensures minimal test time increase and no at-speed coverage loss on the core interface, offering an elegant solution for soft cores, and thus enabling significant SOC test cost reductions.

1. INTRODUCTION

In a manner consistent to the SOC design style, a divide and conquer based testing scheme is crucial in attaining test cost reductions for SOCs. Such an approach delivers benefits that address test time and data volume concerns, IR drop or test power issues, and capacity limitation of ATPG and fault simulation tools. A key component in a core-based SOC test approach is *core isolation*, which is a mechanism that enables independent test of cores. Through a proper core isolation mechanism, core tests can be individually and independently generated, which can be subsequently applied on SOC platform, even if these cores are deeply embedded in SOC logic with reduced controllability and observability of core I/Os. Such a capability also facilitates the theme of test reuse, which proves to be beneficial when cores are reused across multiple SOCs, as recurring cost of test generation for these cores is minimized.

The amount of core structural information provided to the SOC integrator may differ depending on the IP considerations of the core vendors. In the case of soft cores, the core logic is fully exposed to the SOC integrator. This is typically the case when the SOC integrator utilizes in-house cores on the SOC, obviating the need for IP protection. The SOC integrator can fully access the core logic, capable of executing a test generation tool and of enhancing core testability via test point insertion. Consequently, soft cores can be conceived as an extension to the conventional IC design.

IP considerations of vendors force them to hide core structural information from the SOC integrator, especially when the core is being delivered to another company. Such a business model gives rise to the utilization of hard cores, which are delivered in a fully encapsulated format. The SOC integrator is therefore incapable of performing test generation due to the inability to access the core logic. To enable the SOC integrator to target the manufacturing defects within the core, the associated core vendor provides also the test vectors to be delivered to the core. The SOC integrator implements an on-chip test access mechanism that ensures the delivery of the test data from the tester to the core I/Os.

Due to the inability of netlist enhancement in the case of hard cores, their isolation can only be ensured via their treatment as black boxes; as recommended by the IEEE 1500 standard [1], the controllability and observability of core I/Os is perfectly restored via the insertion of multiplexers around hard cores on every core I/O. These multiplexers are controlled by and observed in test-only registers that are also inserted for isolation purposes. While isolation challenge for hard cores is overcome by this technique, area cost imposed by the multiplexers and test-only registers, and performance penalty induced by the multiplexer delay inserted on functional paths ensue. Test time and data volume increase due to the additional test bits to be inserted into the isolating test-only registers is also a costly aspect of such an invasive isolation approach. Furthermore, this approach prevents the at-speed test of synchronous paths/interfaces that cross core boundaries, or requires a top-off run in non-isolated mode, which can be a serious limitation if such interfaces are large in size. This type of an approach also suffers from elevated levels of power dissipation in functional mode and routing congestion, all due to the additional test only registers inserted into the design.

While such a pessimistic and costly approach is crucial in the case of hard cores, a cost effective isolation mechanism based on a judicious analysis of core logic can be implemented for soft cores, as core structural information is available. In this paper, we propose an isolation technique for soft cores wherein existing functional registers are reused for isolating the core. The proposed technique identifies a minimal number of these registers to perfectly isolate the core, minimizing the test time and data volume penalty of core isolation. The proposed core isolation technique imposes negligible area overhead and is non-invasive, as it keeps intact the functional paths that traverse core I/Os. The proposed methodology delivers an elegant and cost effective solution to the challenging problem of SOC test.

The remainder of the paper is organized as follows. An overview of the literature is provided in section 2, while section 3 provides an outline of SOC test basics with emphasized focus on core isolation and its consequences. We present the proposed register partitioning based core isolation technique in section 4, while register capture manipulation techniques that can be utilized to accompany the proposed isolation approach are outlined in section 5. Sections 6 and 7 provide experimental results and conclusions, respectively.

978-3-9810801-2-4/DATE07 © 2007 EDAA

2. PREVIOUS WORK

In recent years, a considerable amount of effort has been expended for testing core-based SOCs. Embedded core test challenges and a survey of the previously published approaches can be found in [2].

Numerous techniques have focused on core isolation techniques. The partial isolation ring technique in [3] proposes the scan of a subset of core I/Os. The test responses of the other cores propagated to the inputs of the core under test are utilized for obtaining the desired core test stimuli, eliminating the scan of a number of core inputs. Similarly, core test responses are utilized to obtain the stimuli for testing the neighboring core(s), enabling partial scan of the core outputs. Performance degradation of this technique can be severe depending on the number of scanned core I/Os. In [4], the utilization of a new type of storage element is proposed for the core outputs; both the observation of the core responses and the justification of the stimuli for testing the neighboring core(s) can be performed at significantly reduced performance cost. However, the technique still suffers from performance degradation due to the multiplexer insertion at the core inputs. A transparency-based test generation technique is proposed for core-based SOCs in [5]. For every core, justification and sensitization paths through other cores are constructed by inserting test points, enabling the translation of core-level stimulus to system-level stimulus. Performance degradation can still occur in this technique due to the insertion of numerous test points. Furthermore, challenges in transparency channel construction limit the applicability of this technique to cores such as digital signal processors and microprocessors. In the SOC test approach proposed in [6], an isolation technique that makes the assumption that core inputs and outputs are directly registered. As this is not the case in typical industrial designs, the benefits offered by this SOC test approach is significantly hampered.

3. SOC TEST OVERVIEW

In a typical test procedure of an SOC with multiple cores, the cores and the glue logic at SOC top-level are tested individually and independently, in order to reap the divide and conquer benefits of core-based test. The mode wherein the internal logic of cores are tested is referred to as *INTEST*, while the glue logic at SOC top-level, which is also referred to as UDL (User Defined Logic), is tested in the *EXTEST* mode. The test of the cores may be effected concurrently, or serially, a decision mostly driven by the test power dissipation concerns, the underlying test access architecture and the allocation of test resources. Typically, EXTEST and INTEST operations are executed in a mutual exclusive manner, however, due to the configuration of the shared isolation mechanism differently in the two modes.

The independence of test operations in SOC cores hinges on a multitude of factors, which include test controller, on-chip clock generation for at-speed testing capabilities, and isolation. These factors can also be generalized as test resources. The decision regarding the test controller and the clock generation circuitry is up to the SOC test integrator to make and has direct implications on the how test operations are to be applied. In one extreme, a single test controller or test clock generation circuitry can be shared among multiple cores and the UDL, while in the other extreme, every core gets a dedicated test controller or a test clock generation circuitry. Other hybrid approaches that fall in between these two extremes can also be employed. These decisions impose certain constraints on core tests, while they also determine the area cost and complexity of the SOC test solution. Sharing a test resource among multiple cores, for instance, reduces the area cost, while the concurrent test of the associated cores should then be effected under the same set of constrains, such as same at-speed test frequencies, or same values for certain test signals.

Isolation is another key feature in attaining independent core test capabilities. With no proper core isolation, the individual core tests must be followed up by a top-up test of the complete chip so as to collect the faults on the core-core or core-UDL interface. This procedure is problematic on multiple fronts. The first reason is that test generation on the complete SOC can be quite time consuming, if possible. The chip may contain too many logic gate instances for the ATPG tool of a certain computational capacity to handle. Also, due to intellectual property considerations of the core vendors, certain core netlists may be unavailable to the SOC integrator, prohibiting the execution of ATPG or fault simulation tools on the complete chip. Other problems associated with the execution of a top-up full chip test is that such a process may beat the purpose of the core-based test theme; test power, test bandwidth, and other test related benefits that the core-based testing approach reaps are degraded by a top-up test performed on the whole SOC treated as a monolithic entity.

Core isolation needs to provide certain features to enable a true core test independence; these features include the restoration of the controllability and the observability loss on the core interfaces. In INTEST mode, for instance, the core primary inputs are all uncontrollable while the core outputs are unobservable, if the core is embedded in the SOC. While certain directly controllable and observable scan cells control the primary inputs of the core and observe the primary outputs of the core, these scan cells reside either in other cores or in UDL. A true independence of the core test dictates that the core test should depend on no scan cells outside the core, necessitating an isolation mechanism that shatters the dependence on external scan cells for controllability and observability.

Analogously, during EXTEST mode, UDL inputs, which are the core outputs, need to be controlled, and the UDL outputs, which are the core inputs, need to be observed with no dependence on any core internal registers. Thus, a proper isolation mechanism should provide controllability of core outputs and observability of core inputs during EXTEST mode, rendering the test of the UDL independently as well.

While an isolation mechanism should implement the capabilities outlined above in order to ensure independent test of cores and UDL, its implications may vary depending on its implementation. Area overhead, SOC performance penalty due to delay insertion on functional paths, test time and data volume penalty, and at-speed coverage loss constitute the potential drawbacks of an isolation mechanism. The utilization I/O scan register and a multiplexer that is inserted on the core functional path on a per core I/O basis [1], for instance, is a costly core isolation approach, due to the associated area cost that consists of as many scan register-multiplexer pairs as the number of all core I/Os. Furthermore, the multiplexers inserted on the functional paths impose a setup time penalty that equals a multiplexer delay, interfering with the functional operation of the SOC timingwise. This approach also imposes test time and data volume penalty as well, since these test only registers need to be accessed in both INTEST and EXTEST modes; the associated test data volume penalty equals twice the number of isolating scan cells, as these non-functional registers are accessed twice. While the scan register provides perfect controllability and observability on core interfaces, all the combinational paths going in and out of the core I/Os are effectively broken into two paths, each getting tested in either the INTEST or the EXTEST mode, if core and UDL tests are to be performed mutually exclusively. Consequently, the at-speed coverage, namely transition or path delay fault coverage, reported on these sub-paths fail to reflect the true at-speed coverages, which is attained only if the whole

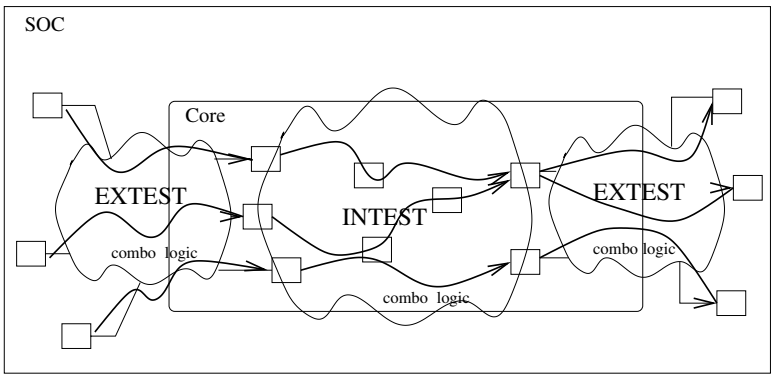

Figure 1: INTEST and EXTEST operations in the proposed scheme

path is tested at-speed all the way from the source registers to the sink registers. Such a mechanism introduces the isolating scan cells that constitute artificial source or sink registers. In order to report true functional at-speed coverage on core interface with this type of an isolation mechanism, a third mode of operation should be introduced wherein the combinational paths that go through core interfaces are tested by keeping the isolating scan cells out of the picture; however, this type of capability imposes further increased area cost, performance penalty and implementation complexity.

4. PROPOSED ISOLATION MECHANISM

4.1 Overview

In this paper, we propose an isolation mechanism that utilizes existing functional registers so as to isolate a core from its environment. Thus, the proposed technique imposes practically negligible area cost and no performance penalty whatsoever. Furthermore, the test of all the combinational paths that go through core I/Os, is effected in EXTEST mode, enabling the reporting of true at-speed coverage on these paths. The cost-effective isolation technique we propose is based on a logic-tracing algorithm that identifies the input and the output interface registers in a core; the remaining registers are classified as core internal registers. The proposed methodology is complemented by one of the capture manipulation techniques that we present in the subsequent section.

Once the proposed technique identifies the input and the output interface registers, the complementary capture manipulation technique is applied so as to ensure that no capture operation is effected in the core input interface registers during INTEST mode and that no capture operation is effected in the core output interface registers during EXTEST mode. During core INTEST mode, the core inputs, which are uncontrollable, are prevented from corrupting any of the registers inside the core. Analogously, during EXTEST mode, the core outputs, which are uncontrollable, are prevented from corrupting any of the registers in UDL logic. In INTEST mode, all the core registers except for the core input interface registers are allowed to capture, while in EXTEST mode, all the UDL registers and the core input interface registers are allowed to capture. A visual illustration of INTEST and EXTEST operations is provided in Figure 1.

In core INTEST mode, the logic between the core input registers and the core output registers is tested, while the core internal registers help apply stimulus and collect fault effects. Core input interface registers and the core internal registers need to be controlled to be able to apply the proper stimuli, while the core output interface registers and the core internal registers need to capture so as to collect fault effects, covering completely the logic that stem from core input interface registers, span core internal registers and end at core output interface registers. In EXTEST mode, on the other hand, the UDL logic, the logic between the core output interface registers and core outputs, and the logic between core inputs and core input interface registers are all tested. Thus, all core output interface registers, and UDL registers should be controlled, and all core input interface registers and UDL registers should be observed, covering completely the logic that stem from core output interface registers, span UDL registers, and end at core input interface registers.

In addition to the employment of a capture manipulation technique that we present in the subsequent section, SOC scan architecture should be designed such that core interface registers are accessed in both INTEST and EXTEST modes; in core INTEST mode, scan shift operation is effected in all the core registers, while in EXTEST mode, scan shift operation is performed not only in UDL registers, but also in core input interface registers as well. This can be achieved in a cost effective manner by constructing dedicated scan chains for the core interface registers, which are accessed in both INTEST and EXTEST modes at the expense of one multiplexer per core interface chain, resulting in negligible area cost. The number of core interface registers is quite important, however, as accessing these registers in both modes contributes to test time and test data volume. The additional test data volume imposed equals the number of core interface registers, as accessing them the second time constitutes the test data volume overhead.

In DfT-friendly cores, wherein core inputs are immediately registered with no fanout or combinational logic in between and where registers directly drive core outputs, core interface registers are trivially identified; thus, the complete core logic is tested during INTEST mode. While SOC cores can be specifically designed to have such built-in isolation capabilities, DfT is rarely a concern for core designers. In a more realistic scenario, core inputs control a cloud of combinational logic before they are registered, and core outputs are controlled by combinational logic driven by core registers. The identification of core interface registers is further complicated, when certain core registers are controlled not only by core inputs but also core internal registers that have no path from/to any core I/O. Registers with an incoming path from a core input, which is by definition a core input interface register, should be prevented from capturing unknown values in INTEST mode. Furthermore, these input interface registers should also be prevented from capturing unknown values in EXTEST mode as well, when there is an incoming path from a core internal register, which is not controlled in EXTEST mode and hence is an unknown source. Consequently, the logic that feeds these kind of input interface registers remain untested. This problematic case

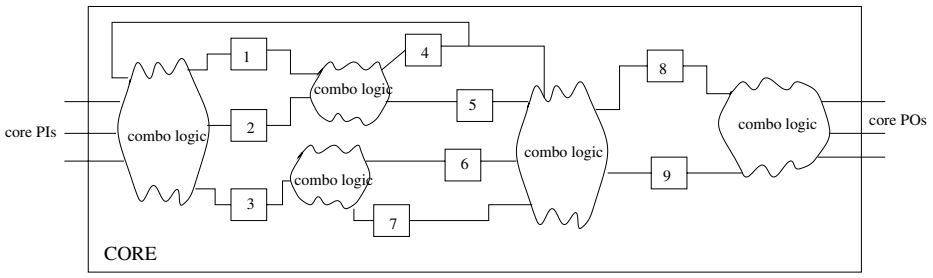

Figure 2: Internal sequential loops that results in isolation problems

is illustrated in figure 2; scan cell 4, which feeds the combinational logic that drives the input interface registers, namely scan cells 1, 2, and 3, results in the isolation problem outlined above. Scan cells 1, 2, and 3 cannot capture in INTEST mode as core PIs are unknown sources, while they cannot capture in EXTEST either as scan cell 4 is an unknown source since it is an internal register. Combinational logic that feeds the scan cells 1, 2, and 3 remain uncovered subsequent to INTEST and EXTEST operations, consequently.

The proposed isolation technique identifies the aforementioned problematic cases by logic tracing, and eliminates these problems by converting certain core internal registers into core interface registers, even if these internal registers do not have any combinational path from/to core I/Os. The optimization criterion consists of the minimization of the number of core interface registers, in order to minimize the increase in test time and data volume.

4.2 Algorithm

The proposed core isolation technique is based on reusing the existing registers in the core and on manipulating their capture operations. The algorithm that we present in this subsection partitions the core registers into three categories in a non-overlapping manner: Input Interface Registers (IIR), Output Interface Registers (OIR), and INTernal registers (INT). The proposed algorithm performs this partitioning such that the no parts of the core logic remain untested subsequent to the application of INTEST and EXTEST modes, while the number of interface registers is minimized. Furthermore, the partitioning is effected via a *depth search first manner*, resulting in a run-time complexity that is *linear* and thus quite efficient computationally.

The partitioning in the core is effected by accounting for the following constraints. During INTEST, data will be scanned into/from IIR, OIR, and INT, while INT and OIR will capture. The UDL, and thus the core PIs will be X-generator, so there should be no combinational path from core PIs to INT and OIR. During EXTEST, data will be scanned in/from top-level (UDL) logic, OIR and IIR, while only UDL and IIR registers will capture. The INT will be X-generators, so there should be no combinational path from INT register outputs to core POs or IIR.

The four-step algorithm below identifies the optimal register partitioning that satisfies the constraints above:

- Trace forward from all core inputs through combinational logic only in a depth-first search manner and mark as IIR all the registers that are reached.
- Trace backwards from all core outputs in a depth-first search manner through combinational logic only and mark as OIR all the registers that are reached and that has not been marked as IIR during the first step.
- Trace backwards from all IIR register data inputs in a depth-first search manner through combinational logic only and mark as OIR all the registers that are reached and that has not been marked as IIR during the first step.
- Mark all the unmarked registers as INT.

While the first, second and the fourth steps of the algorithm are straightforward, the third step of the algorithm ensures that the problematic case mentioned in the previous subsection, namely the failure to test parts of the core logic subsequent to INTEST and EXTEST modes, is not encountered. This step ensures that the incoming paths into the IIRs come from only the core inputs, which are controlled by the UDL registers, and from other core interface registers. Both the UDL registers and core interface registers are controllable in EXTEST mode, perfectly covering the combinational logic feeding the IIR registers. This step can also be perceived as a conversion step, wherein the problematic core internal registers are converted into interface registers in order to eliminate paths from INT registers to IIRs. It is very critical to note, however, that these problematic registers can be converted into IIR or OIR for the elimination of any INT to IIR combinational paths; their conversion into IIR would create additional INT to IIR paths, necessitating more INT to IIR conversions, and thus increasing the number of interface registers. In order to keep the interface register set size minimal, problematic INT registers are converted into OIR registers.

Figure 3 illustrates the execution of the proposed algorithm on an example; register 4, which is an internal register with no combinational path from/to any core I/O, is converted into OIR. The combinational logic that feeds registers 1, 2, and 3 becomes perfectly controllable in EXTEST mode, consequently.

5. CAPTURE MANIPULATION TECHNIQUES

Multiple approaches exist for preventing capture of unknown values. These approaches may have varying implications that include area cost, performance penalty, test time and data volume increase, and at-speed coverage loss. One of these approaches can be selected and employed to complement the proposed register partitioning technique that we presented in the previous section. These approaches are presented below.

5.1 Gating the data-path:

Unknown sources can be blocked by inserting logic gates on the functional data-path, gating these problematic paths to known or fully controllable values. IEEE 1500 standard recommends such a scheme where multiplexers block the unknown sources; one multiplexer and at least one test register per core I/O is necessitated for isolating the core during test. This is a costly approach, due to the associated area overhead, performance degradation, test time increase and at-speed coverage loss. A simpler data-path gating approach consists of inserting gates with controlling values right in front of the data input

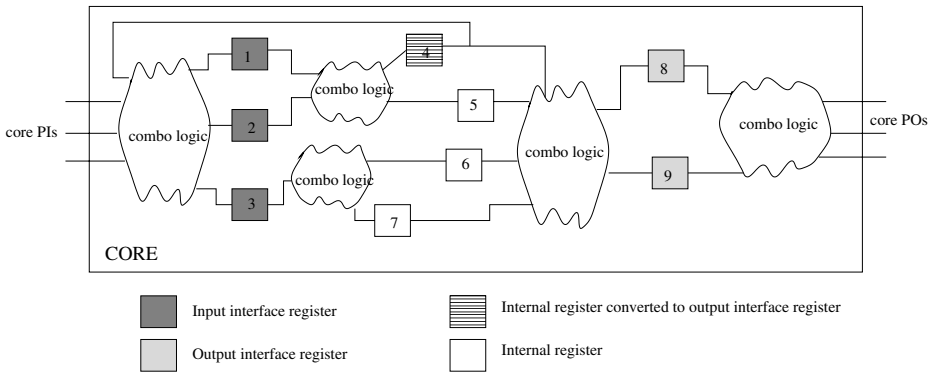

Figure 3: Proposed isolation technique: partitioning registers

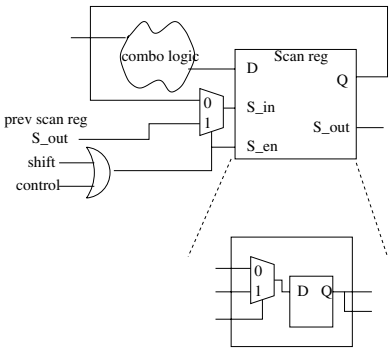

Figure 4: Capture prevention via recirculating scan regs

of the interface registers; by asserting the controlling values of these gates, the interface registers capture known values.

5.2 Gating the clock path:

The prevention of capture operation in a register can be achieved by gating the clock path, and thus suppressing the clock that feeds the register. In order to couple this approach with the proposed isolation technique, the clock of the IIRs can be disabled during capture in INTEST mode, while the clock of the OIRs can be disabled in EXTEST mode. The cost associated with this technique is as many clock gating cells as necessary to ensure the control of all the IIR and OIR clocks. A drawback of this approach is the at-speed coverage loss in the cloud of logic driven by the IIR in INTEST mode, if launch-off-capture scheme is employed as the at-speed testing scheme; as the IIRs are not clocked during capture, no transition can be triggered from these registers, rendering the outgoing paths at-speed untestable. Analogously, at-speed coverage loss is suffered in EXTEST mode in the combinational logic driven by the core OIR. If launch-off-shift technique is employed, however, this technique can be used with no at-speed coverage loss.

5.3 Gating the scan enable signal and the use of re-circulating multiplexers:

Another approach for disabling capture operation in a register consists of gating the shift signal and of feeding the scan register output back to the register [7] as illustrated in figure 4. As a scan cell has a built-in multiplexer whose select line is controlled by the scan enable input of the scan cell, the insertion of another multiplexer on the scan-in input helps attain this capability. During shift, the scan cell selects the scan-in data input, while during capture, the register captures its own output, preserving its state, rather than capturing its functional data input, which may be an unknown source. The control signal that drives the OR gate in the figure determines whether the scan register re-circulates or it captures the data input; for IIRs, this control signal can be tied to the INTEST mode signal, while for OIRs, it can be tied to the EXTEST mode signal.

The cost associated with this technique is one multiplexer per interface register. It should be noted that one OR gate can be shared among all the IIRs, while another is shared for all the OIRs. As multiplexers are inserted on the scan path rather than the functional path, no performance penalty is imposed whatsoever by this capture prevention technique.

As in the clock path gating scheme, this approach too may suffer from at-speed coverage loss, as the preservation of register state during capture prohibits the launch of a transition from this register in a launch-off-capture testing approach. This drawback can easily be eliminated however, by feeding the inverted output of the scan register; during capture, the scan register toggles rather than preserving its state, always launching a transition.

5.4 Gating the scan enable signal to "always high":

Another capture prevention approach consists of gating high the scan enable input of the registers to be isolated. During capture, the register shifts instead, preventing the capture of unknown values coming in from the data input. The scan enable signal feeding the interface registers can be ORed with the INTEST or with the EXTEST signal in order to ensure that the associated interface registers capture the scan out of the preceding registers in the scan chains.

The cost associated with this approach is negligibly small; a single OR gate for all the IIRs, and another OR gate for all the OIRs constitute the area overhead. Furthermore, no performance penalty or no at-speed coverage loss is suffered whatsoever. The only potential caveat imposed by this approach may be that for especially higher speed cores, timing must be met from the scan-out pin of one scan cell to the scan-in pin of the succeeding one on the chains that contains the interface registers. Such a constraint may impose stronger drivers for the scan-out ports of scan cells.

6. EXPERIMENTAL RESULTS

We have implemented the proposed isolation technique in C++ coding language, and applied on several industrial core benchmarks. In this section we not only present the results of the register partitioning based approach that we propose, but we furthermore provide a comparison against the widely utilized isolation technique recommended by the IEEE 1500 [1] approach.

Core	Total regs	Proposed Isolation Technique					IEEE 1500 recommended isolation	
		IIR	OIR	INT->OIR	Total interface regs	TDV penalty(%)	Scan cell-MUX pair	TDV penalty(%)
A	12826	1018	695	937	2650	20.7	1834	28.6
B	2687	343	39	82	464	17.3	439	32.7
C	5920	728	385	226	1339	22.6	1013	34.2
D	26107	561	341	184	1086	4.2	967	7.4
E	25562	559	243	106	908	3.6	727	5.7
F	74840	965	516	282	1763	2.4	1284	3.4

Table 1: Comparison of the proposed isolation technique with the IEEE 1500 recommended solution

We present the results in table 1. The second column denotes the total number of registers in the core, while the number of input interface registers, output interface registers and the internal registers that need to be converted into output interface registers are provided in the third, fourth and the fifth columns; the sixth column shows the total number of core interface registers identified by the proposed algorithm. The eighth column presents the number of scan cell-multiplexer pairs if the core isolation is to be performed as recommended by IEEE 1500; these numbers are identical to the total number of core I/Os. The seventh and the ninth columns denote the test data volume increase in percentage per pattern due to isolation. In the proposed scheme, this number is simply the ratio of core interface registers to all core registers, as the interface registers are to be accessed *again* in EXTEST mode in addition to the INTEST mode. For the IEEE 1500 recommended scheme, the test data volume increase is computed as twice the ratio of isolating scan cells to all registers, as these test-only non-functional registers are to be accessed both in INTEST and EXTEST modes. In the proposed scheme, the 2 factor is missing, as the interface registers are existing functional registers that are to be accessed in INTEST just like any other register in the core; the cost here is their additional access in EXTEST mode.

As can be seen from the table, the test data volume increase for the IEEE 1500 approach can be quite significant, a point that has been overlooked in the past. This problem can become severe, especially for I/O-heave circuits, where the penalty exceeds 30%. The proposed methodology, on the other hand, can successfully address this issue, as the associated test data volume increase remains much below those of IEEE 1500. Another significant limitation of IEEE 1500, namely the area cost of one register and one multiplexer per core I/O, is significantly alleviated by the proposed approach, wherein only a few gates need to be inserted; this is truly a non-invasive approach given that the proper capture manipulation technique accompanies the propose isolation approach. Furthermore, all these benefits can be reaped with no sacrifice in test quality, while significant reduction in test time and data is attained compared to the alternative isolation approaches.

The test data volume and test time cost of the proposed isolation technique remains below those of the IEEE 1500 recommended isolation for all the test cases. Furthermore, the area cost of the proposed scheme is negligible, especially if the "gating scan enable signal to high" capture manipulation is utilized in conjunction, while the area cost of the 1500 recommended solution is one scan cell and one multiplexer per core I/O. The percentage increase in the number of registers in the core equals half of the test data volume penalty percentage numbers, which are provided in the ninth column for the 1500 recommended isolation. There is no register increase in the proposed solution. Finally, the proposed methodology impose no timing penalty and no at-speed coverage loss, as opposed to the IEEE 1500 recommended isolation technique.

7. CONCLUSION

We propose a non-intrusive core isolation technique that enables the independent test of SOC cores cost effectively. The proposed methodology is based on partitioning the core registers, thus identifying the input and the output interface registers. Certain core internal registers that are problematic for isolation purposes are also converted into interface registers, ensuring that no combinational paths exist from core internal registers to the input interface registers. By coupling with one of the capture manipulation techniques that we also present in this paper, core isolation can be successfully effected; core interface registers shield the core logic from its environment while the core is being tested, and the environment from the core while the environment is being tested.

The proposed algorithm is not only quite efficient computationally, it is optimal as well, since it identifies a minimal set of core interface registers; the number of internal registers converted into interface registers is minimal. The test data volume and test time cost of the proposed technique is thus minimized.

As the proposed isolation approach is based on the utilization of existing functional registers in the core, the associated area cost is negligible. A few multiplexers suffice to provide access to the scan chains that contain the interface registers, constituting the area cost of the proposed scheme. Furthermore, with the careful selection of the accompanying capture manipulation technique, performance penalty and at-speed coverage loss concerns can all be eliminated.

The area, performance, test cost, and quality benefits of the proposed methodology can be reaped for soft cores, especially when these issues are exacerbated for high speed cores. The test of SOC cores can thus be performed independently through the elegant and cost effective solution that we propose in this paper.

8. REFERENCES

[1] IEEE Std 1500-2005 , "IEEE Standard Testability Method for Embedded Core-based Integrated Circuits".

[2] Y. Zorian, E. J. Marinissen and S. Dey, "Testing Embedded-Core Based System Chips", in *ITC*, pp. 130–143, 1998.

[3] N. A. Touba and B. Pouya, "Using Partial Isolation Rings to Test Core-Based Designs", *IEEE Design and Test of Computers*, pp. 52–59, October 1997.

[4] S. Bhatia, T. Gheewala and P. Varma, "A Unifying Methodology for Intellectual Property and Custom Logic Testing", in *ITC*, pp. 639–648, 1996.

[5] I. Ghosh, N. K. Jha and S. Dey, "A Low Overhead Design for Testability and Test Generation Technique for Core-Based Systems-on-a-Chip", *IEEE TCAD*, vol. 18, n. 11, pp. 1661–1676, November 1999.

[6] J. Remmers, M. Villalba, and R. Fisette, "Hierarchical DFT methodology - a case study", in *ITC*, pp. 847–856, 2004.

[7] S. Pateras, "Achieving At-speed Structural Test", *IEEE Design and Test of Computers*, pp. 26–33, September-October 2003.