A Bypass Scheme for Core-Based System Fault Testing[†]

M. Nourani[‡] and C. Papachristou[§]

Abstract

We present a global design for test methodology for testing a core-based system in its entirety. This is achieved by introducing a "bypass" mode for each core by which the data can be transferred from a core input port to the output port without interfering the core circuitry itself. The interconnections are thoroughly tested since they are used to propagate test data (patterns or signatures) in the system. The system is modeled as a directed weighted graph in which the core accessibility is solved as a shortest path problem.

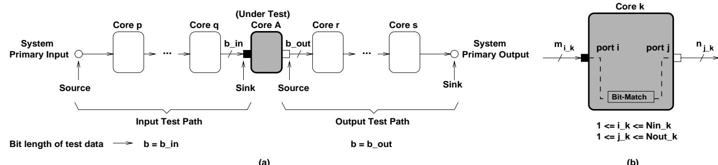

Figure 1: Test paths: (a) Input/output test paths; (b) A typical core in test paths.

1. Introduction

Design and test of core-based systems is a very important and challenging problem facing the semiconductor industry for the next several years. A major difficulty concerns accessibility of embedded cores from the I/O terminals of the system. The basic technique suggested by many R&D groups is to access each embedded core for testing in isolation from the others [KoWa97, BhGV96, ImRa90]. However, there are a number of disadvantages with the isolation approach. Isolation testing does not test the system-on-chip as a whole, for example it does not address the testing of interconnects and interfaces between cores. It does not consider the interaction of cores for testing such as the effect of testing one core on surrounding cores, or vice versa (the surrounding cores also may affect the core under test). Implementing protection safeguards will be very costly increasing the test overhead of the isolation method even further [ToPo97].

There are a number of test methods that apply to ASIC cores based on Design for Test (DFT) techniques, e.g. Scan, BIST, ScanBIST, Test points, or without DFT using precomputed testing. In isolation method, a global BIST controller is usually employed to test schedule cores with embedded BIST structures to shorten the test time. A test bus has also been proposed to affect accessibility.

In this work our basic goal is to test the system as a whole, this means testing the cores themselves as well as testing their interface. We first define the "bypass" mode for each core by which the data can be transferred from a core input port to the output port without interfering the core circuitry itself. Then, we model the core-port accessibility problem as finding the shortest path in a directed weighted graph and minimize the testing time by overlapping the time consumed to access paths.

2. Core Access Using Bypass Mode

The overall objective is to use the existing wires and topology of the system to establish a path to carry test data between two test points in the system, called *source* and *sink*. In core testing, there are two types of test paths, that is *input test path* to access core input ports from system primary inputs and *output test path* to access core output ports from the system primary outputs. Figure 1(a) shows a core under test (*Core A*) and these two paths and their corresponding source and sink. It also shows a general view to establish a route between two test points using the existing wires. In Figure 1(b) the blowup picture of a core in input/output test path (*Core k*) is shown. We symbolically showed that the inputs are bypassed to the output without interfering the core circuitry which are used in the normal mode. The basic idea of having the bypass mode for each core is to have an independent route around a core to carry *test data* (e.g. predefined test patterns or core signatures) between $port_i$ (m_{i_k} bit wide) and $port_j$ (n_{j_k} bit wide) of that core (*Core k*).

Our goal is to establish the shortest path (fastest route) to carry the packets of test data between source and sink. Note that by this formulation, accessibility of the core inputs from system primary inputs and of the core outputs from system primary outputs, are similar problems, i.e. identifying the shortest path between source and sink. The important benefit of the bypass is to take advantage of the existing connections among the cores and the existing wires to transfer multi-bit data from source to sink. This, in the worst case, will be equivalent to the conventional scan which transfers the patterns serially using a separate routing in the system.

2.1 Packetizing Technique for Data Transfer

Figure 1(a) clearly shows that the bit width of inputs/outputs of cores change in a path between two test points. This requires sort of bit matching. Let's assume that we need to transfer a b bit pattern between source and sink. In general, to transfer b bit test data from $port_i$ (m_{i_k} bit) to $port_j$ (n_{j_k} bit) of *Core k* we need to packetize data (to match the available bit width) and send it in several iterations. For example, a core with a $m = 4$ bit input port bypasses $b = 16$ bit test data in 4 iterations. We assume that data packetization and transfer is synchronized with the system clock.

The first type of the bit-match circuit to assemble a larger pattern from different packets of data consists of $\lceil \frac{n}{m} \rceil$ stages of cascaded m-bit registers (e.g. D flip flops) controlled by the same clock. The circuit, which is like a shift register bank of m bits, has a serial-to-parallel (S/P) behavior whose worst case (in terms of time) corresponds to $m = 1$ – equivalent to the traditional scan-in discipline. The second type of the bit-match circuit to disassemble a large pattern to different packets of

[†]This work was partially supported by the Semiconductor Research Corporation (SRC) under Contract No. DJ-527.

[‡]Department of Electrical and Computer Engineering, University of Tehran, Tehran, Iran.

[§]Department of Computer Engineering, Case Western Reserve Univ., Cleveland, Ohio.

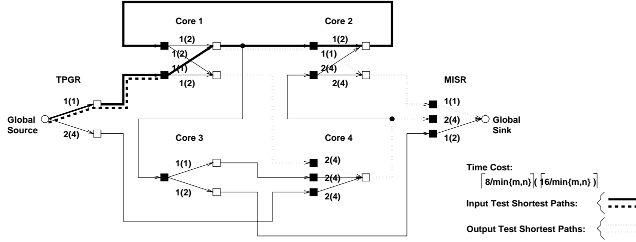

Figure 2: Four input/output shortest paths for *Core 1*.

data consists of n stages of parallel $\lceil \frac{m}{n} \rceil \times 1$ one-bit multiplexors and 1-bit registers (e.g. D flip flops) controlled by the same clock. The circuit has a parallel-to-serial (P/S) behavior whose worst case (in terms of time) corresponds to $n = 1$ – equivalent to the traditional scan-out discipline. Briefly, to bypass b-bit data from an m-bit input port to an n-bit output port of a core, the time cost value is: $\lceil \frac{b}{min\{m,n\}} \rceil$ cycles.

3. Graph Modeling and Core Testing

The shortest path between any two points (called source and sink) reflects the fastest route to transfer packets of data between those two points. From testing perspective, with such model we can find the fastest route to transfer test data (predefined or random pattern) from the system primary inputs to any of the core input ports or to transfer test data (signature) from any of the core output ports to the system primary output.

Equation $\lceil \frac{b}{min\{m,n\}} \rceil$ defines the cost associated with bypassing data from $Port_i$ to $Port_j$ of a core. So, in our graph modeling, a node corresponds to a *port* and an edge corresponds to the interconnects between ports or the bypass possibilities. Those edges reflecting the bypass choices form a bipartite subgraph for each core whose cost (weight) will be determined based on the above equation. In our work, we have used Dijkstra algorithm to find the shortest path which is a greedy algorithm providing an exact solution with computational complexity of $O(|E| + |V|log|V|) \leq O(n^2)$ where $|V|$ and $|E|$ are total number of nodes and edges in the graph, respectively.

As an example, we applied the Dijkstra algorithm to the graph of a system, made of four cores with different ports and bit widths, under test. In Figure 2, we have shown the shortest paths for *Core 1* only.

After finding all the shortest paths for all core input/output ports, we will perform the bypass scheduling for each core. In this work, we have employed a simple As Soon As Possible scheduling method and consider scheduling the cores in order. More sophisticated scheduling methods to relax this two (ASAP and fixed ordering) can be pursued for higher performance. Finally, using this path scheduling method we combine these path schedules to overlap the test activities and minimize the test time as much as possible.

4. Experimental Results

In this section, we demonstrate our approach using a system made of four cores. The circuits have been synthesized from high level descriptions using the SYNTEST synthesis system [HPCN92]. Logic level synthesis is done using the ASIC Synthesizer from the COMPASS Design Automation suite of tools with a 0.8-micron CMOS library [Comp93]. Fault coverage curves are found for the resulting logic level circuits using AT&T's GENTEST fault simulator [Gent93].

Since standard core benchmarks have not been introduced yet we decided to design ourselves four example circuits as cores, all with eight bit wide datapaths. The first core (*Core 2*) evaluates a third degree polynomial. Our other examples implement three high level synthesis benchmarks: a differential equation solver (*Core 3*) and the Facet example (*Core 1*) [GDWL92], and a fifth order elliptical filter (*Core 4*) [KuWK85]. Note that each core consists of the datapath and controller both are made testable using BIST applied by SYNTEST in integrated fashion [HPCN92][NoCP97].

	Full Scan Test		Structural Test	
	Parameter	Value	Parameter	Value
Test Overhead [Transistor]	Input	5192	Core Control	1691
	Output	4484	Bit Match	3548
	Test Controller	1045	Test Controller	6062
Test Time Per Pattern [Cycle]	Core 1	57	Core 1 (Overlapped)	16
	Core 2	101	Core 2 (Overlapped)	15
	Core 3	154	Core 3 (Overlapped)	18
	Core 4	189	Core 4 (Overlapped)	30
	One Iteration	501	One Iteration	34
Overall Statistics	Fault Coverage	88.44%	Fault Coverage	91.43%
	Test Overhead [Transistor]	10721	Test Overhead [Transistor]	11301
	Test Time [Cycle]	57363	Test Time [Cycle]	9180

Table 1: Comparison between scan and our approach

We then put these fours together and obtained the fault simulation of the whole system based on "Scan" and our "Structural" test. Table 1 summarizes the fault coverage statistics for these two methods.

The lower fault coverage of Scan is expected because that method can not capture the interconnect faults. Our proposed Structural test approach requires slightly more test circuitry (about 5.4%) than Full Scan mainly due to its test controller. Overall, the Structural test overhead is 23.18% of the system cost in this example. Using our Structural test approach, the fault coverage has increased by almost 3% mainly due to detecting the interconnect faults. The real advantage of our Structural method is the test time since by bypass scheduling we overlap transferring test data between test points using the existing interconnections. This resulted in almost 84% test time reduction compared to Scan.

References

[BhGV96] S. Bhatia, T. Gheewala and P. Varma, "A Unifying Methodology for Intellectual Property and Custom Logic Testing," Proc. of ITC-96 Conf., 1996.

[Comp93] Compass Design Automation, "User Manuals for COMPASS VLSI V8R4.4," Compass Design Automation, Inc., 1993.

[Gent93] AT&T, "User Manuals for GENTEST_S 2.0," AT&T Bell Laboratories, 1993.

[GDWL92] D. Gajski, N. Dutt, A. Wu, and S. Lin, High-Level Synthesis: Introduction to Chip and System Design, Kluwer Academic Publishers, 1992.

[HPCN92] H. Harmanani, C. Papachristou, S. Chiu and M. Nourani, "SYNTEST: An Environment for System-Level Design for Test," Proc. of EURO-DAC 92, 1992.

[ImRa90] "V. Immaneni, S. Raman, "Direct Access Test Scheme – Design of Block and Core Cells for Embedded ASICs," Proc. of ITC-90 Conf., 1990.

[KoWa97] B. Koenemann and K. Wagner, "TestSockets: A Framework for System-On-Chip Design," White Paper, Logic Vision Inc., 1997.

[KuWK85] S. Kung, H. Whitehouse and T. Kailath, VLSI and Modern Signal Processing, Prentice-Hall, 1985.

[NoCP97] M. Nourani, J. Carletta and C. Papachristou, "A Scheme for Integrated Controller-Datapath Fault Testing," Proc. of DAC-97 Conf., 1997.

[ToPo97] N. Touba, B. Pouya, "Testing Embedded Cores Using Partial Isolation Rings," Proc. of VTS-97 Conf., 1997.