

Test Optimization of Bus-Structured SoCs Using Embedded Processor

M. H. Tehranipour, M. Nourani
Dept. of EE
The Univ. of Texas at Dallas
Richardson, TX 75083
{mht021000,nourani}@utdallas.edu

S. M. Fakhraie
Dept. of ECE
The Univ. of Tehran
Tehran 14399, Iran
fakhraii@chamran.ut.ac.ir

C. A. Papachristou
Dept. of EECS
Case Western Reserve Univ.
Cleveland, OH 44106
cap@eeecs.cwru.edu

Abstract

Embedded processors are now widely used in system-on-chips. This paper presents an optimization technique for testing a bus-structured system using an embedded processor. We present a systematic approach for test access mechanism that allows processor to access all cores with minimum overhead. We show an ILP formulation to minimize the test schedule. The method requires negligible overhead but provides great flexibility in terms of access mechanism and future reuse.

1 Introduction

System chips are increasingly designed by embedding large pre-designed and pre-verified modules called *cores*. Examples of cores are processors, DSPs, memories, and communication modules of various kinds [1] [2]. Testing core-based systems is a major challenge. The main reason is that the accessibility of the cores and blocks is greatly reduced. Additionally, the system designer might have a limited knowledge of the core internals due to the protection of the intellectual property (IP). Several factors such as fault coverage, test time, performance and area overhead need to be considered to determine an effective test strategy when integrating cores in a system chip.

Embedded test integrates multiple disciplines: DFT features, BIST pattern sources and sinks, precision and high-speed timing for at-speed test, test support for many different core types (logic, memory, processors, and analog), and capabilities for diagnosis and debug. With embedded test, the on-chip test data generation reduces the volume of external interaction which can be customized per core type. Also, the on-chip test and diagnostic data compression reduces ATE data logging and ATE speed requirements [3].

The problem with external test pattern based strategies is that they typically require large test data volume and need complex dynamic test control protocols to get stimulus to cores, receive responses from cores and set up the necessary control to execute the test on a core. A promising strategy that minimizes such requirements is BIST. All known benefits of BIST come from the fact that BIST embeds test into the circuit under test. It generates and evaluates the test patterns on-chip [4]. This can be performed as a software-based test that provides test procedures (to generate deterministic and random data) by a processor core for testing itself or other cores in the system chip [5] [6].

The cores are connected by on-chip interconnects. To connect the cores, manufacturers often use bus-structured systems (BSS) because they are easy to build and efficient to implement. A major difficulty concerns accessibility of embedded cores from the I/O terminals of the system [7]. The proposed methodology describes the accessibility of cores by an on-chip processor. There are three main approaches to deliver test patterns to an individual core and observing core responses. One approach uses extra test circuitry around the core to isolate the core during test [8]. The second approach uses a transparency or bypass mode for embedded cores to reduce the problem to one of finding paths from the system inputs to the core inputs and from the core outputs to the system outputs [9]. The third approach is based on test bus or a flexible TESTRAIL around the cores to transfer test data [10].

Given a set of cores and their individual test information to be integrated into a system-chip, our goal is to devise a test access mechanism whereby a complete test of the system can be accomplished quickly and inexpensively using embedded processor. In this work, we take the advantage of all access elements (e.g. MUXes, controllable buffers, bypass routes, tristate ports etc.), generally called bridges [7]. The proposed test access mechanism is formulated as an ILP problem to minimize the overall test time and also scheduling the test process.

Embedded processors can be utilized to run software routines for test pattern generation and response evaluation [5] [6] [11]. One of the major problem is how processor access to all of the cores under test. We propose a test access mechanism for accessing all of the cores in a BSS by processor. Also, an ILP formulation has been implemented to obtain a scheduling with minimum time steps. We assume that each core in the system-chip has a predesigned wrapper. It is utilized for accessing the cores by processor via the buses.

In our approach, processor tests other embedded cores. A direct downloading mechanism, e.g. direct memory access (DMA), is used to store test programs in program memory (PM) before the test session begins. The testing process is effectively done at-speed or near at-speed since the entire testing process is done inside the system. Additionally, the test controller is almost non-existent because the processor uses its normal-mode controller (i.e. instruction decoder) to execute the test program. More importantly, since the test mechanism is specified in software the method is very flexible.

The rest of this paper is organized as follows. Section 2 describes the software-based system chip testing. Using processor for SoC testing and Developing a test access mechanism

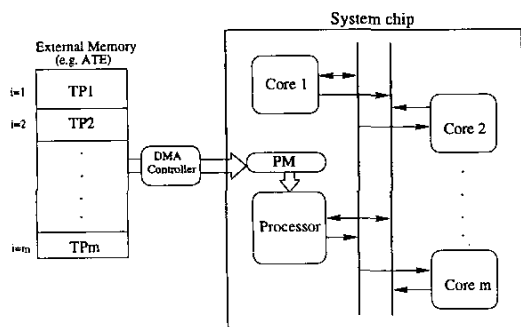


Figure 1. System chip software-based testing.

(TAM) are described in this section. TAM ILP formulation is shown in Section 3. Experimental results are presented in Section 4. Finally, concluding remarks are in Section 5.

2 Software-Based BSS Testing

2.1 Using Embedded Processor

We show a general view of testing a BSS using an embedded processor. Figure 1 shows the general concept of a software-based test for a BSS. As shown, there is a test program (TP) for each embedded core. In this paper, we do not talk about test program generation. Some researchers have reported methodologies for test program generation [6] [11]. Each test program consists of four procedures: Pattern Generation Procedure (PGP), Test Delivery Procedure (TDP), Signature Generation Procedure (SGP) and Comparison Procedure (CMP). For example, for testing $Core_i$, first TP_i is loaded into PM of the processor by a DMA and then processor executes the loaded TP_i .

After execution of TP_i by processor, it will generate test patterns (random or deterministic) and applies to $Core_i$ by accessing its input ports. Processor will also read test responses and it needs to access output ports of $Core_i$. Test responses will be compressed and then generated signatures are compared with predefined fault-free signatures. Final results are reported to the primary outputs of system chip by processor. As seen, TAM is one of the major challenges in system chip test. Our test methodology needs a TAM based on processor core. In the next section, we develop a systematic TAM that processor is able to access embedded cores.

2.2 Test Access Architectural Model

We call any controllable component that allows two points to be connected in an SoC a *bridge*. Bridges are used to connect the processor to all of the buses in the BSS. As mentioned, processor tests all cores in the system chip therefore, it must access all cores. Usually processor already has access to some of the buses in normal mode. Our formulation shows how minimum additional bridges for test mode, should be chosen to allow accessing all buses/cores. Processor also controls

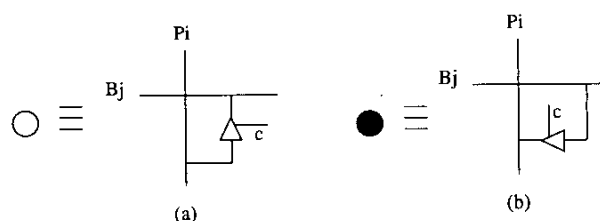


Figure 2. Symbolic representation of bridges (a) $P_i \rightarrow B_j$, (b) $B_j \leftarrow P_i$.

the bridges for transmitting and receiving patterns. Additional bridges are only used in the test mode and thus they do not affect the overall performance in the normal mode.

We first identify the cores environments which comprises all the core input/output connections from/to buses. Figure 2 shows the connection of the bridges on the system chip. Figure 2.(a) shows a bridge that transfers data from P_i to B_j . Figure 2.(b) depicts a bridge that transfers data from B_j to P_i . The related symbols are shown in the figure. Signal c is bridge control (e.g. enable) and will be controlled by processor. Figure 3 shows a general view of our test access mechanism in a small BSS.

Processor is able to send data on the buses toward cores and then receives the test results from the cores. In our approach the ILP formulation determines the number of bridges needed for testing purpose. We assume that the processor has the largest bit-width of data line. Therefore, it can send patterns on the less and equal bit-width buses. This is not a true restriction as by using extra hardware parallel-to-serial and serial-to-parallel bridges can be designed. Typical bridges of this nature are shown in [7]. There are two phases in the test process:

1. **Apply phase:** Processor sends data on the buses. The data can be sent simultaneously on the several buses for multiple cores. Several cores can use identical data on one bus if they have a connection on that bus. Here, we assume that all of the cores can be tested with BIST and they need the random patterns. The formulation can be modified if there are some cores that use deterministic test patterns. In the random-based testing, fanout is a constraint that shows how many buses are able to receive patterns at the same time in apply phase.
2. **Receive phase:** In this phase, processor receives test results from the cores sequentially.

Figure 4 shows how data will be applied and read by the processor for a small SoC. As shown in apply phase, P_{10} (output port of the processor) applies patterns on the buses (B_1 , B_2 , B_3 , and B_4). As seen in apply phase, patterns can be applied on P_1 and P_8 through bus B_1 simultaneously. Applying patterns on P_4 and P_6 are also performed from bus B_2 . In general, apply phase in random-based testing can be performed in one time step without fanout consideration. But it may be important and we consider the fanout constraint in the

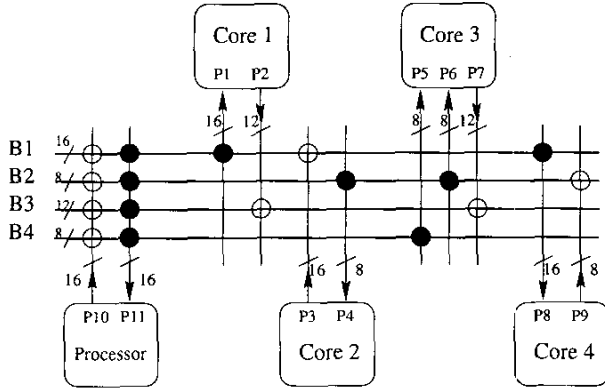


Figure 3. A small example of a system chip.

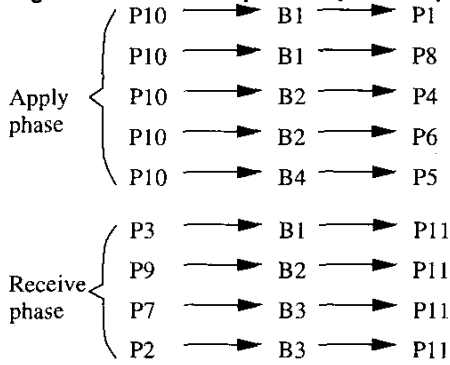


Figure 4. Apply and receive sessions.

ILP formulation of designed system. Another factor which should be considered in the ILP formulation is using the same bus in apply phase and receive phase. In this case, the bus must be used in different time steps to avoid conflict.

Processor sequentially reads the test results from the cores therefore, the number of time steps in receive phase depends on the number of output ports on the cores. This increases the number of time steps of the system chip testing using processor but it results in at-speed or near at-speed testing and also it reduces cost significantly. Our ILP formulation finds the minimum time steps needed for both apply and receive phases.

3 ILP Formulation

Some bridges are added for using only in test mode. The other bridges are used in test and normal mode. Processor has already some connections with some of the buses and the formulation finds additional bridges needed. In the test mode, processor controls all (normal and test mode) bridges.

In this section, we show how to optimize the access of the cores and scheduling of the test process by an ILP formulation. This formulation finds two set of access paths for each core which eventually form the TAM; one from processor output to the core inputs to be used for delivering test patterns, and one from the core outputs to the processor input to be used to observe core response. The constraints in the ILP formulation

are:

1. The number of fanout that test designer is allowed to use will be defined by the system designer.
2. Apply and receive phases can not use a bus in the same time step. Also, processor should receive responses only from one bus in each time step in receive phase.
3. Receiving test response should be performed after applying test patterns plus execution time for each core, i.e. $T_{receive} \geq T_{apply} + d_k$, which d_k is execution time for the core k .

Assume that the cores (not processor) in system chip have totally n input/output ports (P_1, P_2, \dots, P_n) that need to be controlled and observed. Also, assume that there are p buses on the system chip (B_1, B_2, \dots, B_p). The buses are the interface between the processor and other cores. The number of cores (not included processor) in system chip is m . The maximum time step is λ that test designer is allowed to put all of the test tasks in apply and receive phases in the scheduling. Suppose i, j, k and l show ports, buses and time step, respectively. Their ranges are $1 \leq i \leq n, 1 \leq j \leq p, 1 \leq k \leq m$ and $1 \leq l \leq \lambda$. Also, a is the maximum number of authorized fanout. x_{il} is the ILP variable:

$$x_{il} = \begin{cases} 1 & \text{if } P_i \text{ is used at time step } l. \\ 0 & \text{otherwise.} \end{cases}$$

Minimize:

$$(i) \quad \sum_{i,l} l \cdot x_{il} \quad \forall i, l \quad (1)$$

Subject to:

$$(i) \quad \sum_{i,l} x_{il} = 1 \quad \forall i, l \quad (2)$$

This uniqueness shows that applying data from processor to each input port and receiving data from each output port by processor is performed in one of the time steps.

$$(ii) \quad \sum_l l \cdot x_{il} - \sum_l l \cdot x_{jl} - d_k \geq 0 \quad (3)$$

Where $i, j = \{1, \dots, n\}$ and $k = \{1, \dots, m\}$. This constraint guarantees that receiving test response from each core is performed d_k time steps after applying test pattern.

$$(iii) \quad \sum_{m=l-d_k+1}^l x_{rm} \leq a \quad (4)$$

Where r is input ports of the cores and $r \subset i$. In Figure 3, $i = \{P_1, P_2, \dots, P_9\}$ and $r = \{P_1, P_4, P_5, P_6, P_8\}$. This inequality shows fanout constraint. It guarantees that at most

Table 1. CPLEX results.

a	Running Example		NDV8401	
	# Bridges	Time Step	# Bridges	Time Step
1	4	6	5	15
2	4	4	5	11
3	4	4	5	10
>4	4	4	5	9

a fanouts are allowed in apply phase to apply patterns to a input ports in parallel.

$$(iv) \sum_{s_j} x_{s_j, l} \leq 1 \quad (5)$$

Where s_j is cores input and output ports that use the same bus B_j as $s_j \subset i$ and $j=\{1, \dots, p\}$. This constraint guarantees that input and output ports (in apply and receive phases) that use the same bus, use the bus in different time steps. In Figure 3, P_1 and P_8 in apply phase and P_3 in receive phase use bus B_1 . P_1 and P_8 which are input ports may use B_1 in the same time step, but they cannot use B_1 with P_3 in the same time step, therefore $s_1=\{P_1, P_8\}$, P_3 . Also, P_2 and P_7 use bus B_3 in receive phase and then $s_3=\{P_2, P_7\}$. These two ports should be read in different time steps.

4 Experimental Results

The ILP formulation is applied on two SoCs: 1) our running example shown in Figure 3 and 2) NDV8401. NDV8401 is a DVD-on-a-chip taken from [12]. The NDV8401 DVD controller is cost effective integrated system for universal DVD player/records. This devices includes eleven cores. For running example, we suppose that processor has already access to B_1 and B_4 from P_{10} and B_2 from P_{11} . It shows that there are three bridges in normal mode.

Table 1 shows the results of running CPLEX [13] on re-unning example and NDV8401. It shows that six time steps are sufficient for all tasks in test process (apply and receive phases) for running example when it is allowed to use only one fanout in apply phase. As shown, by increasing the number of fanouts, required time steps are reduced for both SoCs. It also reduces the number of required control lines. Table 1 also shows that the number of required test bridges is four for running example and five or NDV8401.

The scheduling obtained from CPLEX for running example for $a=3$ is shown in Figure 5. ILP formulation also finds the location of required test bridges. Scheduling in Figure 5 shows that two 16-bit bridges from B_1 and B_4 to P_{11} , one 8-bit bridge from P_{10} to B_2 and one 12-bit bridge from B_3 to P_{11} should be added in test mode.

5 Conclusion

This paper presents testing a bus-structured system using an embedded processor. The main part of work is the ILP for-

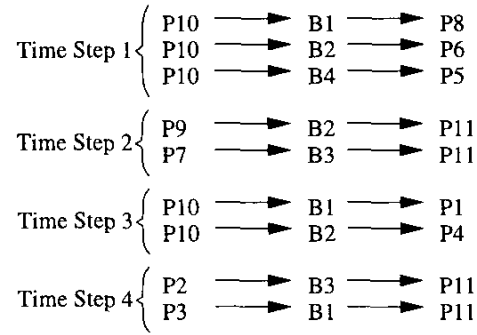


Figure 5. Scheduling of test process for running example.

mulation for test access mechanism. It shows how processor can access all of the cores in the system chip with minimum time steps. It also finds the number and location of required bridges for test purpose and time scheduling of processor tasks in applying test patterns and receiving test responses.

References

- [1] R. K. Gupta and Y. Zorian, "Introducing Core-Based System Design," IEEE Design & Test of Computers, vol. 14, no. 4, pp. 15-25, 1997.
- [2] Y. Zorian, E. J. Marinissen and S. Dey, "Testing Embedded-Core Based System Chips," in Proc. Int. Test Conf. (ITC'98), pp. 130-143, 1998.
- [3] Y. Zorian, S. Dey and M. Rodgers, "Test of Future System-on-Chip," in Proc. Int. Conf. on Computer-Aided Design (ICCAD'00), pp. 392-398, 2000.
- [4] M. Bushnell and V. Agrawal, *Essentials of Electronic Testing*. Kluwer, 2000.
- [5] C. Papachristou, F. Martin and M. Nourani, "Microprocessor Based Testing for Core-Based System on Chip," in Proc. Design Automation Conference, pp. 586-591, 1999.
- [6] R. Rajsuman, "Testing a System-on-a-Chip with Embedded Microprocessor," in Proc. Int. Test Conf. (ITC'99), pp. 499-508, 1999.
- [7] M. Nourani and C. Papachristou, "An ILP Formulation to Optimize Test Access Mechanism in System-on-Chip Testing," in Proc. Int. Test Conf. (ITC'00), pp. 902-910, Oct. 2000.
- [8] V. Immaneni and S. Raman, "Direct Access Test Scheme- Design of Block and Core Cells for Embedded ASICs," in Proc. Int. Test Conf. (ITC'90), pp. 488-492, Oct. 1990.
- [9] M. Nourani and C. Papachristou, "Structural Fault Testing of Embedded Cores Using Pipelining," in *Journal of Electronic Testing: Theory and Applications (JETTA)*, pp. 129-144, Oct. 1999.
- [10] E. Marinissen, R. Arendsen, G. Bos, H. Dingemans, M. Lousberg and C. Wouters, "A Structured and Scalable Mechanism for Test Access Mechanism for Test Access to Embedded Reusable Cores," in Proc. Int. Test Conf. (ITC'98), pp. 284-293, Oct. 1998.
- [11] W. Lai and K. Cheng, "Instruction-Level DFT for Testing Processor and IP Cores in System-on-a-Chip," in Proc. Design Automation Conf. (DAC'01), pp. 59-64, 2001.
- [12] National Semiconductor, Inc., <http://www.national.com>.
- [13] CPLEX user manual, www.ilog.com