Hardware/Software Co-testing of Embedded Memories in Complex SOCs

Bai Hong Fang, Qiang Xu and Nicola Nicolici
Department of Electrical and Computer Engineering
McMaster University, Hamilton, ON L8S 4K1, Canada
Email: fangbh@mcmaster.ca, xuqiang@grads.ece.mcmaster.ca, nicola@ece.mcmaster.ca

Abstract

A novel approach for testing embedded memories in complex systems-on-a-chip (SOCs) is presented. The proposed solution aims to balance the usage of the existing on-chip resources and dedicated design for test (DFT) hardware such that the functional power constraints are not exceeded during test while trading-off the testing time against DFT area and performance overhead. The suitability of software-centric and hardware-centric approaches for embedded memory testing is examined and to combine the advantages of both directions, a new built-in self-test (BIST)-based method, called hardware/software co-testing, *is introduced. The proposed solution is programmable, scalable and guarantees low routing overhead.*

1 Introduction

It is anticipated that embedded memories will account for 95% of the silicon area in complex SOCs by 2016 [7]. Low cost/high quality testing of these heterogeneous SOCs, comprising a large number of embedded memories, poses a great challenge to the test community. The emerging solutions should address limited bandwidth, at-speed test for performance validation, long testing times, on-chip DFT area overhead and performance penalty, and excessive power dissipation, which may cause manufacturing yield and reliability concerns [10].

There are two main approaches for testing embedded memories: direct access and BIST [4]. On the one hand, direct access of the embedded memory cores from the limited number of I/O pins needs a high-performance automatic test equipment (ATE), as well as very long testing time. Thus, direct access is infeasible, in particular for complex SOC devices where transistor to pin ratio is high. On the other hand, memory BIST (MBIST) provides at-speed and high-bandwidth access to the embedded memory cores, and it only needs a low cost ATE to initialize the tests and to inspect the final results. However, although BIST is state-of-the-art technology for embedded memory testing, unless carefully designed, it may induce excessive power, in addition to performance and area overhead.

ICCAD'03, November 11-13, 2003, San Jose, California, USA.

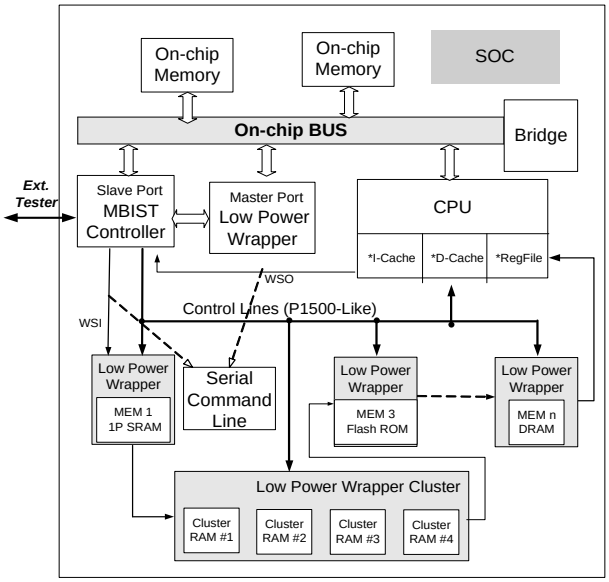
1.1 Motivation for New Approaches for Power-Conscious Hardware/Software Co-testing

The embedded memory cores in an SOC can be divided into two groups: bus-connected memories (BCMs) and non-bus-connected memories (NBCMs). Although all the embedded memory cores can be tested by adding dedicated memory BIST wrappers, the high area overhead of BIST circuitry, as well as the performance penalty caused by intrusive DFT hardware, may prove to be the main drawback of this approach. Therefore, since a complex SOC usually contains one or more processing elements (e.g., microprocessors), which use on-chip system busses to communicate with the memory cores, reusing the existing on-chip resources for testing the embedded memories can lower the overhead associated with a high number of dedicated MBIST wrappers. In [12], a test methodology for testing SOCs using a microprocessor core was presented. A *software-centric* embedded memory testing approach *(i.e., memory testing is undertaken by an on-chip processing element which reuses the given functional resources and interconnect topology)*, eliminates the overhead caused by BIST wrappers, however it requires a significantly longer testing time than the existing *hardware-centric* approaches *(i.e., memory testing is done using dedicated DFT, e.g., BIST hardware and interconnect resources)* [1, 9, 18], which can add to the overall cost of SOC test. The main purpose of this research is to develop an SOC test solution, which maintains the benefits of both *software-centric* and *hardware-centric* approaches, while satisfying the power constraints during test. The importance of power dissipation during test and its relationship to test cost parameters such as testing time and DFT area overhead are discussed next.

Power dissipation is becoming a key challenge for the deep submicron complementary metal-oxide semiconductor (CMOS) digital integrated circuits. Placing more and more functions on a silicon die has resulted in higher power/heat densities, which impose stringent constraints on packaging and thermal management in order to preserve performance and reliability. While low power design techniques have been employed for more than two decades, the

latest International Technology Roadmap for Semiconductors (ITRS) [7] anticipates that power will be limited more by system level cooling and test constraints than packaging. This is because, if packaging and thermal management parameters (e.g., heat sinks) are determined only based on the functional operating conditions, the high activity during test will affect both manufacturing yield and reliability [10].

On the one hand, when testing bus-connected memories in an SOC the power constraints are easily satisfied, however the main drawback lies into the serialization of the test schedule which leads to excessive testing time. To address this problem, a processor-programmable solution was proposed in [14], where a BIST circuit was inserted between the processor and the system bus. Although it reduces the testing time, this solution can affect the overall SOC performance, since the BIST circuitry may increase the delay on the critical path. Therefore, new approaches need to be sought to address the problems associated with software-centric approaches. On the other hand, since not all the embedded memories in an SOC are connected to a bus system, hardware-centric memory BIST approaches are necessary for all the non-bus-connected memories. While the previous *hardware-centric* approaches [1, 9, 18] have tackled the core-level aspects of memory BIST, to further reduce the overhead at the system-level as well as to reduce the test control complexity associated with complex and heterogeneous SOCs, distributed solutions are necessary. For example, a distributed MBIST architecture was proposed in [3]. This architecture has relatively low area overhead, however its main drawback lies in the control mechanism, which configures all the memory wrappers to run identical test commands in each test session. This implies that memories running different test algorithms (e.g., heterogeneous memories) cannot be tested simultaneously, thus decreasing test control flexibility and increasing testing time. Furthermore, the testing time for each test session is dominated by the largest memory, which may lead to prohibitively long testing time under power constraints. To address power-constrained embedded memory testing, one effective solution is to limit the number of concurrent memory blocks using test scheduling under power constraints. The BIST architecture proposed in [3], can be adapted to this solution, however it will lead to prohibitively long testing time under power constraints when the large memory blocks dominate the time spent for each test session [3]. Hence, new flexible BIST architectures need to be investigated, which will guarantee both low area and control complexity, as well as high test concurrency under given power constraints. To achieve this, a control mechanism must be provided to convert non-partitioned testing[3] to partitioned testing with run to completion [5], as well as to lower both the area overhead and the routing congestion associated with the test control for non-bus-connected memories.



* : I-Cache, D-Cache, and RegFile are all wrapped separately

Figure 1. New Memory BIST Architecture

To test all the memories in an SOC, a mixture of software-centric and hardware-centric self-test approaches is researched in this paper. By balancing the test access/functions for bus-connected memories between functional resources and dedicated DFT hardware, one can easily explore various test configurations with different testing time and area overhead under the given power constraints. This testability trade-off exploration is achieved by a test scheduling engine, which is common to both bus-connected and non-bus-connected memories. In summary the proposed solution comprises the following features:

(i). Reusable IP core for hardware/software co-testing;

(ii). Applied to both bus-connected memories (BCMs) and non-bus-connected memories (NBCMs);

(iii). High flexibility for power-constrained test scheduling;

(iv). Shared BIST controller for heterogeneous memories;

(v). Serial wrapper control for low routing overhead;

The remainder of this paper is organized as follows. The proposed power-constrained programmable MBIST architecture, facilitating *hardware/software co-testing* is described in Section 2. Section 3 details the software structure and test schedule organization. A new test scheduling algorithm is described in Section 4. Section 5 gives the experimental results and Section 6 concludes the paper.

2 Proposed MBIST Architecture for Hardware/Software Co-testing

This section details a new MBIST architecture for *hardware/software co-testing, which can be defined as the process of consciously partitioning the test access, test application and test control functions for embedded memories between dedicated DFT hardware and the available*

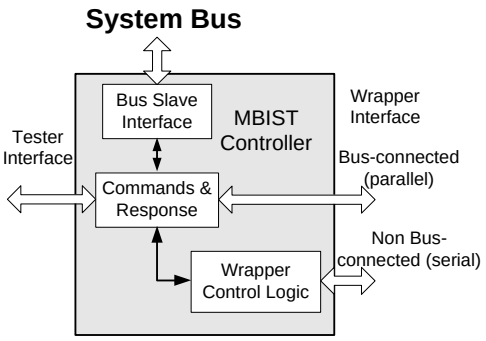


Figure 2. Memory BIST Controller

on-chip functional resources. The proposed MBIST architecture (see Figure 1) consists of two components: the programmable MBIST controller and the MBIST wrappers (detailed in the following two subsections). There are two types of wrappers: wrappers connected to the system bus to test BCMs (can be shared by all the memories on the same bus), and individual wrappers for each NBCM (or cluster of identical NBCMs).

To test BCMs, unlike the approach reported in [14], the proposed solution uses the standard bus interface to exchange data between the central processing unit (CPU) and the MBIST controller, and hence it can affect only the bus performance. Furthermore, if the critical path of an SOC is not on the system bus (which is a realistic case in practice), the presented solution will not influence the SOC performance. In addition, by using a standard bus interface, the proposed MBIST module can be reused as an soft IP core.

When testing NBCMs, to overcome the disadvantage of supporting only non-partitioned test scheduling, as in [3], the proposed MBIST controller has full controllability for each wrapper. This new MBIST architecture supports partitioned testing with run to completion[5], which gives more flexibility to power-constrained test scheduling. In addition, by running most of the non-time-consuming tasks, such as fetch and decode of test commands, in the processing unit using software, the BIST area overhead of the MBIST controller will be reduced.

2.1 Novel Programmable MBIST Controller

One of the distinctive features of the proposed solution is a shared BIST controller for heterogeneous memories. As shown in Figure 2, to communicate with CPU or ATE, the MBIST controller has a bus slave interface to the on-chip CPU and a serial interface to the off-chip ATE. If one wishes to test memories using the ATE, the controller can simply bypass the test data to (from) the serial scan chain, which links all the memory wrappers together. To test BCMs, the MBIST controller passes parallel commands to (from) the BCM wrapper through the parallel interface between them. To test all the other NBCMs, the MBIST controller has to do parallel/serial conversion, send commands to each wrapper, and then do serial/parallel conversion for the results received from NBCM wrappers before passing them to the CPU. Another key feature of the proposed MBIST architecture is the interconnect mechanism between the memories, wrappers and controller. Due to its simple, yet powerful, interface, by programming the BIST controller, any hardware/software co-testing schedule can be implemented.

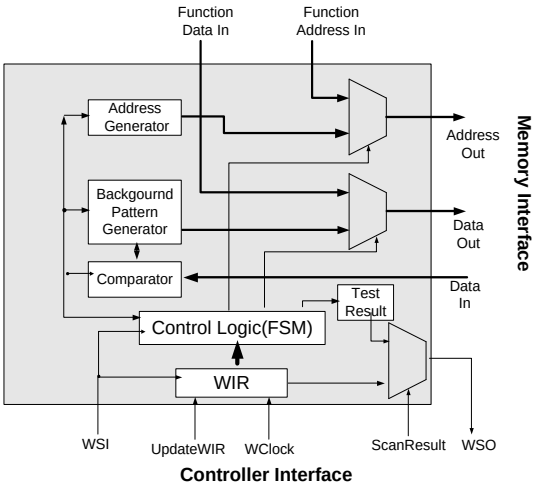


Figure 3. NBCM Wrapper

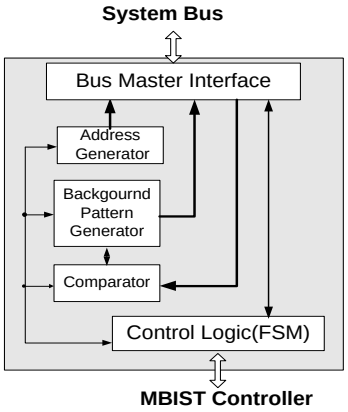


Figure 4. BCM Wrapper

2.2 MBIST Wrappers

Both BCM and NBCM wrappers (as shown in Figure 3 and 4) have a full hardware implementation to run March-based memory testing algorithms [15]. This includes: address generator, background pattern generator, response analyzer, and a finite state machine (FSM) to interpret commands received from the MBIST controller and generate appropriate control sequence to perform memory testing.

For NBCMs, each embedded memory must be wrapped with dedicated DFT hardware (Figure 3). To reduce the wire delay and routing congestion, the wrapper must be placed close to the memory core. To control all NBCM wrappers in an SOC with less routing overhead, the interconnect between the MBIST controller and NBCM wrappers is IEEE P1500-like [11], since it has shared control

lines for all the wrappers and a serial scan chain for sending commands (receiving results) to (from) the MBIST wrappers. All the test commands and test responses are scanned in and out using the WSI and WSO ports. The wrapper instruction register (WIR) [11] stores the command received from the MBIST controller and using WClock and UpdateWIR the new command is updated. The ScanResult signal controls the scan chain to scan in WIR commands or scan out test results. The comparator checks the memory output data for any mismatch. If an error occurs, then BIST will stop and will set the error register. The controller can then scan out the erroneous address and background pattern for diagnosis. Using this control mechanism, the MBIST controller can control each wrapper separately. This implies that different test algorithms can run simultaneously in separate wrappers. Therefore, heterogeneous memories can be connected and tested concurrently using the same BIST controller. To further reduce wrapper and controller area, identical memory blocks are wrapped as a memory cluster. All the memory blocks in one cluster (up to 31) share the same wrapper. The test program first enables all the memory blocks in the cluster and, only if an error occurs, after the controller has finished all the test scheduling tasks, it will test each memory block individually to spot the exact faulty memory .

The BCM wrapper, on the other hand, uses the bus interface to test all the memories connected to the bus. Since the testing process is serialized due to the common test access resource (functional bus), we need a single BCM wrapper, which has a parallel connection to the MBIST controller (see Figure 4). Note, both BCM and NBCM wrappers can run different March elements received from the MBIST controller, which implies that they can run different March algorithms. This is very useful for diagnosis purposes as described in the next section. To further reduce the complexity of manufacturing test, each wrapper is built-in with a default March algorithm, which can be activated using a single command.

3 Software Implementation

The BIST architecture described in the previous section can facilitate hardware/software co-testing for a given test schedule, as described in Section 4. This section details the software implementation necessary to handle the test control flow. First, the test commands are generated using power-constrained test scheduling algorithm (see Section 4) and loaded to a *test memory* (e.g., I-cache) which was pretested using ATE. The test software then reads commands from the test memory and sends data (read responses) to (from) the MBIST controller. Since the time-consuming tasks (i.e., on-chip generation of March tests) are conducted by dedicated BIST hardware, software functions are used only for test control which insignificantly affects the overall testing time. In the following the pseudo-code is given.

Memory Testing Software

1. **Test** CPU and Bus;
2. **Test** Cache memories and Register files using ATE;
3. **Load** Test program to I-Cache, test data to D-Cache;
6. **Do** {
7. **Read** commands from test memory;
8. **Send** commands to MBIST controller;
9. **Wait** for response;
10. **if** error **then break**;
11. }**Until** all the scheduled test commands are completed;

The software implementation supports two modes of operation: manufacturing test mode and built-in self-diagnosis (BISD) mode. In the manufacturing test mode, only one self-test command is needed for each wrapper, which will automatically run the embedded default March algorithm and set the fail/pass bits after completing the testing process. When in the diagnosis mode, the CPU can send any March element supported by the wrapper to run different March algorithms and read back the results (address, background pattern) for further diagnosis purposes. In addition to aiding fault diagnosis, this is a very suitable mechanism to find the best March algorithm for a new technology or to increase the test quality when the current default March algorithm cannot achieve the targeted defect coverage.

4 Test Scheduling

In this section, the test scheduling engine, which is also used to explore different hardware/software co-testing configurations is introduced. Since the SOC bus architecture will affect the way how the BCMs are accessed during test, it will also affect the test schedule and ultimately the testing time. If only one bus master can access the bus, all the BCMs must to be tested sequentially. If the SOC has several large BCMs, this SOC bus architecture will adversely affect the testing time. To address this drawback, one can use alternative bus architectures, such as multi-layered AMBA bus [2], to which several bus masters can be attached and non-shared slave components can be accessed concurrently. By adding one or more BCM wrappers (masters), this architecture supports more flexible test scheduling, thus reducing the testing time. However, if no multi-layered SOC bus architectures are present for the native mode of execution, another solution would be to wrap some BCMs, which have a critical impact on testing time, with dedicated wrappers and test them as NBCMs. Due to our practical validation setup, for the test scheduling algorithm presented in this section and in our experimental results, we have considered the second option (i.e., to boost test concurrency we transform some BCMs to NBCMs), however, note, the same trade-off exploration engine can be used if multi-layered SOC bus architectures are employed.

4.1 Problem Formulation

Recently a test scheduling formulation and algorithm have been reported in [16] only for NBCM. The test scheduling problem for the proposed architecture is a generalization of the one presented in [16], since in our case we have both BCMs and NBCMs. For NBCMs the constraints are the same as in [16]. However, to account for the specific test features of BCMs, each memory connected to a bus will have different resource conflict relationship and values for testing time: one when it is wrapped with dedicated wrapper (i.e., when it is transformed to a NBCM to increase test concurrency) and one when it time-shares the wrapper connected as a master port to the SOC bus architecture. The scheduling algorithm automatically chooses which BCM will be transformed to NBCM, thus exploring various test configurations with different testing time and area overhead.

The parameters used in the algorithm are listed below:

P_{max} power constraint during test;
n_m total number of BCMs and NBCMs;
n_b total number of busses;
for $i \in 1, 2, ..., n_m$
p_i maximum power dissipation for memory i;
lw_i testing time for memory i when wrapped;
luw_i testing time for memory i when unwrapped;
s_i scheduled test start time for memory i;
l_i scheduled testing time for memory i;

Formulation:

Objective: Minimize $T_{max} = MAX(s_i + l_i)$

Subject to: $\sum p_i \leq P_{max}$ where i are the memories currently under test;

4.2 Test Scheduling Algorithm

Since test scheduling is proven to be an NP-complete problem [5], in this section we propose a greedy heuristic to deal with complex SOCs comprising hundreds of memories.

The Algorithm *Mem_Schedule* takes the test parameters of each memory (p_i, lw_i, luw_i) and the power constraint (P_{max}) as inputs, and it outputs the test schedule $T_{schedule}$ and the test method for each memory core M_{test} (i.e., wrapped or unwrapped). Note, this algorithm also increases the number of unwrapped BCMs, without affecting the testing time. The test of each memory is represented as a rectangle whose width is the testing time and whose height is power dissipation. For BCMs two rectangles with different testing time are necessary (one for each test method), while for NBCMs one rectangle representation is sufficient. If two BCMs on the same single-master bus are unwrapped, then they cannot be tested at the same time due to bus contention, and we call these two memories incompatible. The proposed algorithm is based on the rectangle packing algorithm *TAM_schedule_optimizer* proposed in [8] for SOC testing.

The algorithm starts by initializing each memory wrapper test method M^i_{test}: all memories are treated as wrapped except those explicitly specified as unwrapped. Then the testing time T^i_{time} of each memory is computed based on its test method (line 2). The currently available power constraint P_{avl} is initialized to P_{max} and the number of unscheduled memories is initialized to the total number of embedded memories (line 3). As long as there is an unscheduled memory, the algorithm first finds a compatible memory m_i with maximum testing time max_{tat}, which does not exceed the power dissipation constraint (line 6). If such a memory m_i exists, it will be scheduled and the available power dissipation constraint will be updated (line 8, 9). If no compatible memories meet the power constraint, we will record the idle power dissipation P_{idle}, update the available power dissipation $P_{avl} = 0$ (line 11) and branch to the end of the schedule of the memory with the minimum end time min_{end}. Afterward we update the new schedule information, including the new schedule begin time, the number of unscheduled memories $N_{unscheduled}$ and available power dissipation P_{avl} (line 12-15). If there is still some idle test time, we will look for the already scheduled memories and check whether we can unwrap them without affecting the testing time (line 16, 17). The algorithm exits when all the memories have finished their schedule. Although the proposed test scheduling algorithm supports partitioned testing with run to completion [5], it can easily be adapted to non-partitioned testing used by other MBIST architectures [3] (in line 12 we must finish the schedule of m_j with max_{end}).

Memory Schedule Algorithm *Mem_Schedule*

INPUT: M, P_{max}
OUTPUT: $T_{schedule}$, M_{test}

```
1. Initialize M^i_test;
2. Compute T^i_time;
3. Initialize P_avl = P_max; N_unscheduled = N_memories
4. while (N_unscheduled != 0) {
5.     if (P_avl > 0) {
6.         find compatible m_i with max_tat and P_i < P_avl;
7.         if (found) {
8.             schedule m_i;
9.             P_avl -= P_m_i;
10.        } else {
11.            P_idle = P_avl; P_avl=0;
.          }
.      } else {
12.        Finish schedule of m_j with min_end;
13.        Update schedule begin time;
14.        N_unscheduled--;
15.        P_avl += P_m_j; P_avl += P_idle
16.        if (idle time exists) {
17.            unWrap scheduled mems if not incompatible;
.        }}}
18. done
```

Table 1. MBIST Area Overhead for LEON SOC

	BCM	NBCM	Total
Memory (μm^2)	4,100,930	2,698,855	6,799,785
Wrapper(μm^2)	14,449	109,248	123,697
Controller (μm^2)	9,684		
BIST Area Overhead (%)	0.35	4.04	1.96

5 Experimental Results

To prove the correctness and the effectiveness of the proposed MBIST architecture, that facilitates hardware/software co-testing, we have implemented and fabricated an SOC platform based on LEON [6]. LEON is an open source core for embedded applications and it has an IEEE-1754 (SPARC V8) [13] compatible integer unit with 8 register windows, 1k I-Cache, 1k D-Cache, and a full implementation of the AMBA-2.0 bus [2].

To estimate and compare different test parameters (e.g., BIST area overhead, testing time, performance penalty, power dissipation) of the proposed solution, a set of experiments have been performed using high-density embedded SRAMs compiled for 0.18μ CMOS technology. In our LEON-based SOC platform, the NBCMs are a cluster of four 512x16 SRAMs, one 1kx32 SRAM and one 4kx32 SRAM, while the BCMs are one 4kx32 SRAM and one 16kx32 SRAM. The wrappers have been configured to implement 9 March elements: *(w), (r), (rw), (rwr), (rww), (rwww), (rwrwr), (rwrwrw), (wwrww)*, which are the building blocks for most of the known March test algorithms [15, 18]. The last March element *(wwrww)* is used to run the word-oriented March C- test (*March-CW* proposed in [17]). All wrappers also implement March-CW algorithm as the default memory test algorithm. Table 1 shows the BIST area overhead for the LEON SOC. By adding a very low area MBIST controller (0.14% in this case), the CPU can control self testing for most of the embedded memories (note, cache memories and the register file are tested using ATE as outlined in Section 3). The MBIST area overhead (control + wrappers) of this LEON SOC is less than 2%. Timing analysis indicates that the critical path is in the CPU core, which means that our approach does not introduce performance degradation. However, the NBCM wrappers do affect the memory access speed.

Table 2 shows the comparison of three different approaches (software-centric, hardware-centric, and programmable BIST core attached to the bus to be used with hardware/software co-testing) for bus connected memories using the LEON platform [6] and applying the March-CW algorithm [17]. As shown in the table, the proposed solution combines the benefits of the other two approaches. It requires low area overhead, it may affect only the bus performance, and it maintains the flexibility of the software-centric approach, while it reduces the testing time significantly, bringing it close to the best possible testing times given by the hardware-centric approach.

Table 2. Different Approach for Testing BCMs.

	Software Centric	Programmable BIST Core	Hardware Centric
Testing Time	7.0	1.4	1.0
BIST Area	0	Low	High
Performance	0	Affect bus speed	Affect mem. speed
Flexibility	Any March	Most March	Fixed

Using the proposed MBIST architecture and the greedy heuristic for test scheduling described in Section 4, Table 3 gives a comparison between partitioned test with run to completion and non-partitioned test scheduling algorithms under power constraints. The memory cores included in this experiment are of different sizes with the number of address lines ranging from 7 to 16 and the word size ranging from 8 to 32. The power dissipation for these cores ranges from 3.5 mW to 50 mW for 100 MHz clock frequency. On the one hand, it can be seen that the maximum testing time of non-partitioned testing is always greater than (or at least equal to) the partitioned testing with run to completion. The difference between the two testing times varies based on the maximum power constraint and memory configurations. For example, the more we relax the constraints, i.e., higher test concurrency can be achieved, the difference in testing time increases, unless the maximum concurrency has been achieved by both algorithms. One the other hand, because there is more idle time in non-partitioned testing, more BCMs can be unwrapped and tested serially using the on-chip bus. This means that for non-partitioned test scheduling we may increase the testing time at the benefit of lower BIST area overhead. It should be noted that the proposed greedy heuristic is very fast (e.g., for 300 memories it takes less than 1 second).

Finally, the trade-off between BIST area overhead and testing time is shown in Table 4. Item *"BCM as NBCM"* means the BCM has a dedicated wrapper and it will be tested as NBCM. To decrease BIST area overhead, i.e., all BCMs are unwrapped, the testing time will be increased. Similarly, to reduce the testing time, most of the BCMs have to be wrapped thus increasing the BIST area overhead. Using the test scheduling engine, as a trade-off exploration tool, is beneficial especially when the testing time for the entire SOC is dominated by the scan testing time of embedded logic cores. In this case, we can explore different hardware/software co-testing configurations until we match the logic cores' testing time with the lowest DFT area required by dedicated MBIST wrappers.

6 Conclusion

This paper has shown that for SOCs comprising tens to hundreds of embedded memories, a new solution called *hardware/software co-testing* can reduce the testing time and area/performance overhead under power constraints.

Table 3. Testing Time (cc) and Wrapped Memories for Different Power Constraints and Test Schedules

Memory Cores	3	7	10	30	50	70	100	150	200	250	300
Maximum Power Dissipation = 250mW											
Partitioned	147456	147456	156672	986112	1299456	1612800	2082816	3382272	4165632	5465088	7944192
Non-Partitioned	184320	147456	156672	1004544	1317888	1631232	2101248	3400704	4184064	5497344	7976448
Difference(%)	20.00	0.00	0.00	1.83	1.40	1.13	0.88	0.54	0.44	0.59	0.40
Unwrapped (part.)	0	0	1	0	0	0	2	3	5	6	7
Unwrapped (non-part.)	0	0	1	2	4	6	9	14	19	23	25
Maximum Power Dissipation = 500mW											
Partitioned	147456	82944	73728	589824	663552	737280	893952	1419264	1704960	2276352	3428352
Non-Partitioned	147456	101376	82944	764928	903168	1032192	1248768	1672704	2022912	2916864	3700224
Difference(%)	0.00	18.18	11.11	22.89	26.53	28.57	28.41	15.15	15.72	21.96	7.35
Unwrapped (part.)	0	0	1	1	1	0	2	0	0	4	1
Unwrapped (non-part.)	0	0	1	3	5	7	10	15	20	25	19
Maximum Power Dissipation = 1000mW											
Partitioned	147456	82944	73728	589824	589824	589824	589824	829440	1105920	1382400	1972224
Non-Partitioned	147456	101376	73728	700416	829440	958464	1124352	1437696	1732608	2045952	2654208
Difference(%)	0.00	18.18	0.00	15.79	28.89	38.46	47.54	42.31	36.17	32.43	25.69
Unwrapped (part.)	0	0	1	0	1	0	0	0	0	2	0
Unwrapped (non-part.)	0	0	1	5	7	9	12	16	21	25	26

References

[1] D. Appello, F. Corno, M. Giovinetto, M. Rebaudengo, and M. Reorda. A P1500 compliant BIST-based approach to embedded RAM diagnosis. In *Proc. IEEE Asian Test Symposium*, pages 97–102, 2001.

[2] ARM Inc. AMBA Specification. http://www.arm.com.

[3] M. L. Bodoni, A. Benso, S. Chiusano, S. D. Carlo, G. D. Natale, and P. Prinetto. An Effective Distributed BIST Architecture for RAMS. In *Proc. IEEE European Test Workshop*, pages 119–124, 2000.

[4] M. Bushnell and V. Agrawal. *Essentials of Electronic Testing*. Kluwer Academic Publishers, 2000.

[5] G. L. Craig, C. R. Kime, and K. K. Saluja. Test scheduling and control for VLSI built-in self-test. *IEEE Transactions on Computers*, 37(9):1099–1109, September 1988.

[6] Gaisler Research. LEON Web Site. http://www.gaisler.com.

[7] International SEMATECH. *The International Technology Roadmap for Semiconductors (ITRS): 2001 Edition*. http://public.itrs.net/Files/2001ITRS/Home.htm, 2001.

[8] V. Iyengar, K. Chakrabarty, and E. Marrinissen. On using rectangle packing for SOC wrapper/TAM co-optimization. In *Proc. IEEE VLSI Test Symposium*, pages 253–258, 2002.

[9] S. Koranne, C. Wouters, T. Waayers, S. Kumar, R. R. Beurze, and G. S. Visweswaran. A P1500 Compliant Programmable BistShell for Embedded Memories. In *Proc. IEEE International Workshop on Memory Technology, Design and Testing*, pages 21–27, 2001.

[10] N. Nicolici and B. M. Al-Hashimi. *Power-Constrained Testing of VLSI Circuits*. Kluwer Academic Publishers, Frontiers in Electronic Testing (FRET) Series, 2003.

[11] P1500 SECT Task Forces. IEEE P1500 Web Site. http://grouper.ieee.org/groups/1500/.

[12] R. Rajsuman. Testing a System-on-a-Chip with Embedded Microprocessor. In *Proceedings IEEE International Test Conference (ITC)*, pages 499–508, Sept. 1999.

Table 4. Testing time (cc) vs. Wrapper Num.

	min.	max.	diff.
Memories=100 Power=250mW			
Testing Time(part.)	2082816	2709504	30.09%
BCM as NBCM(part.)	18/20	0/20	18
Testing Time(non-part.)	2101248	2912256	38.60%
BCM as NBCM(non-part.)	11/20	0/20	11
Memories=300 Power=250mW			
Testing Time(part.)	7944192	10672128	34.34%
BCM as NBCM(part.)	53/60	0/60	53
Testing Time(non-part.)	7976448	10994688	37.84%
BCM as NBCM(non-part.)	35/60	0/60	35

[13] Sparc International Inc. Sparc V8 standard. http://www.sparc.org/standards/V8.pdf.

[14] C. Tsai and C. Wu. Processor-Programmable Memory BIST for BUS-Connected Embedded Memories. In *Proc. ASP-DAC*, pages 325–330, 2001.

[15] A. J. van de Goor. *Testing Semiconductor Memories: Theory and Practice*. 1998.

[16] C. W. Wang, J. R. Huang, Y. F. Lin, K. L. Cheng, C. T. Huang, C. W. Wu, and Y. L. Lin. Test Scheduling of BISTed Memory Cores for SOC. In *Proc. IEEE Asian Test Symposium*, pages 356–361, 2002.

[17] C. W. Wang, C. F. Wu, J. F. Li, C. W. Wu, T. Teng, K. Chiu, and H. P. Lin. A Built-In Self-Test and Self-Diagnosis Scheme for Embedded SRAM. In *Proc. IEEE Asian Test Symposium*, pages 45–50, 2000.

[18] W. L. Wang, K. J. Lee, and J. F. Wang. An On-Chip March Pattern Generator for Testing Embedded Memory Cores. *IEEE Transactions on VLSI Systems*, 9(5):730–735, 2001.