CASP: Concurrent Autonomous Chip Self-Test Using Stored Test Patterns

Yanjing Li, Stanford University
Samy Makar, CSwitch Corporation
Subhasish Mitra, Stanford University

Abstract

*CASP, **C**oncurrent **A**utonomous chip self-test using **S**tored test **P**atterns, is a special kind of self-test where a system tests itself concurrently during normal operation without any downtime visible to the **end-user.** CASP consists of two ideas: 1. Storage of very thorough test patterns in non-volatile memory; and, 2. Architectural and system-level support for autonomous testing of one or more cores in a multi-core system using stored patterns, concurrently with normal system operation, without bringing down the entire system. CASP enables design of robust systems with built-in features for circuit failure prediction, error detection, self-diagnosis and self-repair. Such systems are necessary to overcome major reliability challenges in scaled-CMOS technologies. Implementation of CASP in the OpenSPARC T1 multi-core processor demonstrates its effectiveness and practicality.*

1. Introduction

CASP is an acronym for **C**oncurrent **A**utonomous chip self-test using **S**tored test **P**atterns. It is a special kind of self-test where a system tests itself concurrently during normal operation without any downtime visible to the **end-user.** The basic idea is to store very thorough test patterns in non-volatile storage, such as hard disks or FLASH memory, and provide architectural and system-level support for testing one or more cores in a multi-core system, while the rest of the system continues to operate normally. CASP enables application of very thorough tests with quantified test coverage, including high-quality scan and functional tests, during normal operation in the field. Extremely thorough tests, some of which may not be applied during production for test cost reasons, may also be applied during system operation using CASP. The main motivation for CASP is to enable robust system design with self-healing capabilities required to overcome major scaled-CMOS reliability challenges such as aging and infant mortality. Major CASP features are:

1. CASP is useful for circuit failure prediction [Agarwal 07], error detection based on periodic, time-triggered or event-triggered self-test [Al-Asaad 98], and self-repair [Bardell 91].
2. CASP applies high-quality test patterns with quantified test coverage (including production test patterns).
3. CASP utilizes already existing on-chip Design-for-Testability (DFT) and test compression features (that are used for production testing purposes).
4. Test patterns can be changed (e.g., through patches) according to application requirements and failure characteristics even after a system is deployed in the field.
5. CASP utilizes major technology trends such as the availability of high-density and low-cost non-volatile storage (mainly off-chip) in future systems, proliferation of multi-core architectures [AMD 07, Azul 07, Cisco 07, IBM 07, Intel 07, nVidia 07, SUN 07], and wide adoption of test compression.
6. CASP is applicable to both datapath and control logic.
7. CASP imposes significantly lower overhead compared to traditional redundancy techniques.

CASP is complementary to and overcomes limitations of many existing on-line testing techniques (details in Sec. 4). Major contributions of this paper are:

1. Introduction of the idea of CASP and demonstration of its practicality and effectiveness;
2. Detailed implementation of special architectural and system features to enable CASP in the open source OpenSPARC T1 chip multi-processor [Sun 06] with eight cores and thirty-two threads;
3. Analysis of trade-offs associated with the design and application of CASP in future systems.

CASP is applicable to a wide range of multi-core systems such as microprocessors (e.g., Intel Core2 Duo, AMD Opteron, and Sun Niagara), networking chips (e.g., 192-core Cisco Metro chip including 4 spares), and GPUs (e.g., the nVidia GeForce 8800 Ultra GPUs with 128 Shader processing units). Such systems with lots of cores are expected to dominate future designs. It is speculated that multi-core systems with thousands of cores may be available in the future [Borkar 07]. We utilize the presence of multiple cores to our advantage for CASP. This paper focuses on the logic part of the processor cores because Built-In-Self-Test (BIST) and self-repair techniques already exist for on-chip memories, e.g., [McNairy 04, Molyneaux 07].

This paper demonstrates CASP using scan test patterns for the following reasons: 1. Scan tests are *de facto* tests with high test coverage; 2. It is possible to automatically generate scan test patterns for a wide variety of test metrics including stuck-at, transition, N-detect, bridging, path delay, and Cadence Encounter Test True-Time delay [Cadence 04]. However, CASP is applicable for functional tests as well.

Section 2 discusses the motivation for CASP. In Sec. 3, we present detailed implementation of CASP in OpenSPARC T1 together with synthesis and test coverage results, and trade-off analysis. Section 4 presents related work, followed by conclusions and future work in Sec. 5.

2. Motivation for CASP

Aging (also called wearout) and infant mortality (also called early-life failures) pose major reliability challenges in scaled-CMOS technologies [Borkar 05, 07]. Conservative design techniques that incorporate speed guardbands are becoming expensive due to aging mechanisms that were benign in the past, but are becoming important in future technologies. Burn-in, which is used to screen infant mortality suspects, is becoming difficult and expensive.

Robust systems with built-in resilience to aging and infant mortality must be designed to overcome these major reliability challenges at very low cost. One way of designing such systems is to employ on-line circuit failure prediction. *Circuit failure prediction* predicts the occurrence of a circuit failure **before** errors actually appear [Agarwal 07]. This is in contrast to traditional error detection where a failure is detected **after** errors appear in system data and states. Circuit failure prediction is effective for aging and early-life failures because

978-3-9810801-3-1/DATE08 © 2008 EDAA

of their gradual nature of degradation. It uses a wide variety of special circuits, unlike traditional process monitors, called sensors, at various locations inside a chip to collect information about various system parameters over time, concurrently during normal system operation and periodic on-line self-test. The collected information is analyzed to predict failures. Effective circuit failure prediction requires:

1. A circuit must be thoroughly exercised.
2. Sensors must be turned on infrequently (e.g., 1-5% of the time) to minimize aging of the sensors themselves, and also to minimize chip-level power penalties.

Hence, on-line testing techniques must satisfy:

1. High test coverage for high-quality on-line tests.
2. Minimal system-level performance impact.
3. No system level downtime visible to the end-user.
4. Minimal hardware cost.
5. No major changes in design and validation flows.

As described in Sec. 3, CASP meets these objectives. It is ideal for circuit failure prediction, but is also useful for self-test-based error detection and diagnosis.

3. CASP for an Open-Source Multi-Core Processor

In this section, we describe architectural implementation and system-level support for CASP in the OpenSPARC T1 processor (Fig. 3.1).

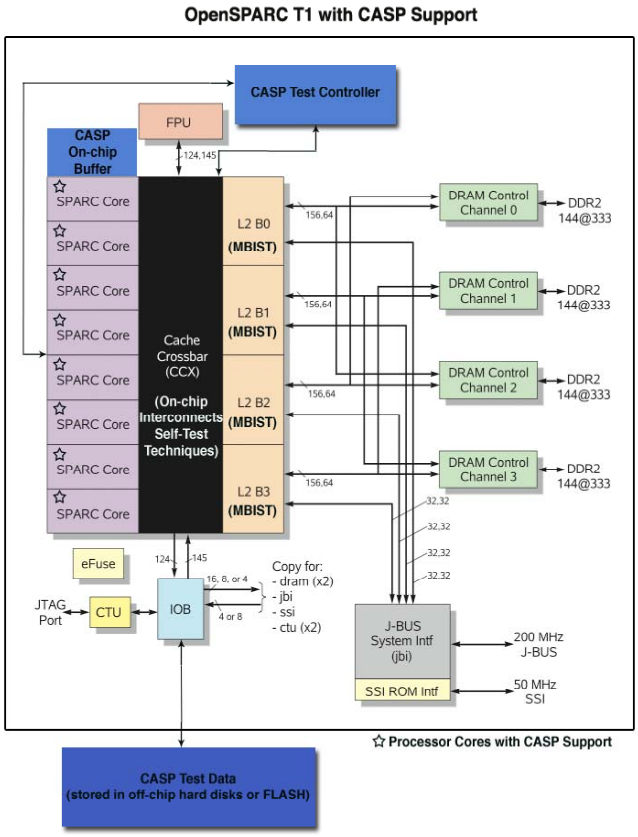

Figure 3.1. OpenSPARC T1 with CASP Support (modified from http://opensparc-t1.sunsource.net)

OpenSPARC T1 is a multi-threaded chip multi-processor with eight SPARC processor cores, each with hardware support for four threads. Each core contains its own instruction and data caches and TLBs, and a single-issue, six-stage pipeline. The eight processor cores communicate with the on-chip unified L2 cache, the floating-point unit (FPU), and the I/O subsystem through a crossbar. The on-chip J-Bus controller provides interconnection between the crossbar and the I/O subsystem. To support CASP in OpenSPARC T1, we implemented a CASP test controller, an on-chip buffer to store a scan test pattern and its corresponding expected response and mask, and necessary architectural features for supporting testing of processor cores during system operation (Fig. 3.1). All test data is stored off-chip in non-volatile storage (hard disks or FLASH memory).

Figure 3.2 presents a high-level overview of CASP for OpenSPARC T1. There are four phases:

1. **Test scheduling:** While the system operates in its normal operating mode, one or more cores may be selected for on-line self-test by the CASP test controller (Sec. 3.1).
2. **Pre-processing:** Execution on the selected core is stalled and the core is temporarily isolated from the rest of the system by stalling the pipeline, disabling communication, saving critical states, and invalidating the L1 data cache (Sec. 3.2).
3. **Perform CASP test:** CASP test controller sets proper signals for applying test patterns and analyzing test responses. Tests are loaded from non-volatile memory, applied to the core under test, and analyzed for failures.
4. **Resume Normal System Operation:** Critical states of the core-under-test are restored, communication is enabled, and the pipeline is restarted (Sec. 3.2).

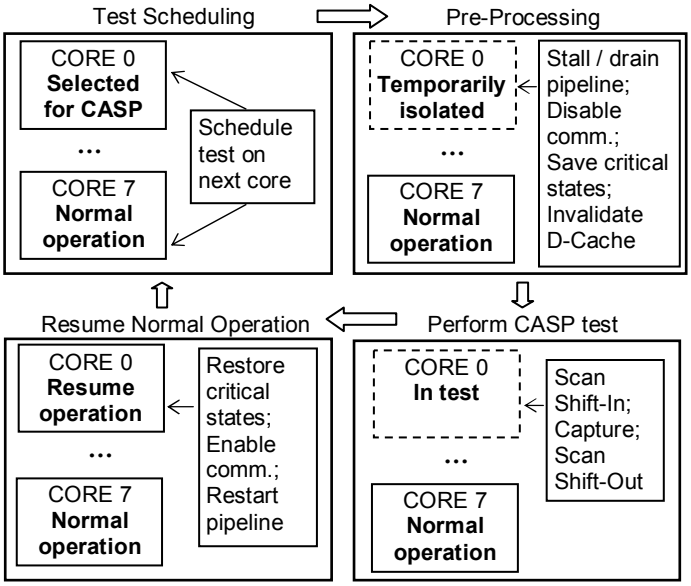

Figure 3.2. CASP implementation in OpenSPARC T1

The following subsections present implementation details. We focus on processor cores of OpenSPARC T1 for the following reasons: 1. They are complex; 2.The FPU can also be tested using CASP; 3. Self-test of on-chip interconnects have been discussed in several recent publications; 4. Memory BIST can be used for on-chip memories including the caches.

3.1. CASP Test Controller and Test Clock Support

CASP requires an on-chip test controller (Fig. 3.3). The test controller is responsible for scheduling self-test in one or more processor cores. Our current test scheduling mechanism simply selects a single processor core at a time in a round-

robin fashion. Optimized test scheduling techniques for minimizing system-level power / performance impact are also possible using higher-level system support, e.g., the operating system or a virtual machine monitor.

In addition to test scheduling, the CASP test controller produces signals to control various CASP operations: 1. Fetch test patterns from the off-chip non-volatile storage to the on-chip buffer; 2. Initiate proper pre-processing of a core before it enters test mode; 3. Perform scan test of the selected processor core with test mode and test clock control signals; and, 4. Compare test responses with expected test responses and bring the core-under-test out of test mode to resume normal operation if the test passes. If the test fails, the test controller can invoke self-correction mechanisms for circuit failure prediction [Agarwal 07], or adopt self-recovery mechanisms available in the system (e.g., rollback in a checkpointed processor). The CASP test controller area is less than 0.01% of the total area of the synthesized design.

System and test clocks are multiplexed by the scan enable signal. To generate a test clock that is sufficiently slow for scan shifting, we use the 150 MHz JBus clock which is in a separate clock domain from the system clock. Control signals from the CASP test controller are synchronized between the system clock and the test clock. Advanced timing tests, e.g., True-Time [Cadence 04], require multiple-speed structural test support, e.g., [Iyengar 06].

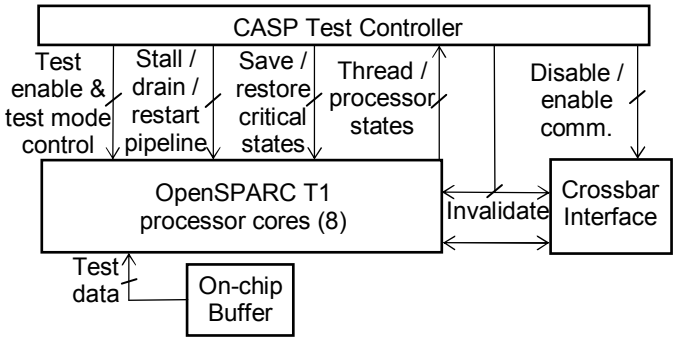

Figure 3.3 CASP Test Controller

3.2 Architectural Support for CASP

As shown in Fig. 3.2, CASP architectural features include: 1. Support for stalling and draining the pipeline, disabling communication, saving critical states, and invalidating the L1 data cache in the pre-processing phase for the selected processor core; and, 2. Support for restoring critical states, enabling communication, and restarting the pipeline after the CASP test completes. For processors with checkpointing, CASP can be implemented with minor modifications. For processors without such support, such as OpenSPARC T1, architectural features for CASP can be introduced with moderate design effort, as described next.

Pipeline Stalling, Resuming, and Draining

Pipeline stalling is already implemented in OpenSPARC T1 to stall a thread when it experiences a long latency instruction such as a load miss. We use this signal to stall all threads of a processor core when the CASP test controller asserts the test mode signal corresponding to that core. The pipeline is resumed when the CASP test controller de-asserts the corresponding test mode signal.

When the stall signal is asserted, there may be outstanding instructions in the pipeline. We allow these instructions to complete before the processor core switches into test mode. To drain the pipeline, states of all four threads are first examined by the CASP test controller to make sure that all instructions are fetched (in the case of instruction misses). When such a check passes, thread states are checked again until the test controller detects that all threads are in idle, halt, or ready state, indicating no outstanding instructions are currently in the pipeline. The CASP test controller ensures that all store instructions have updated the lower-level memory by checking that the store buffer is empty.

L1 Data Cache Invalidation

During CASP operation, other cores may execute store instructions that require invalidation of some entries in the L1 data cache of the core-under-test, which will not respond. To preserve cache coherence, we invalidate all L1-data cache entries of the core-under-test by clearing the valid bit array. The invalidation guarantees correctness since the L2 cache is inclusive. OpenSPARC T1 uses an L2 cache directory that shadows L1 cache tags to manage cache coherence. We also clear the valid bit array for the corresponding entries in the L2 cache directory.

Communication Disable and Enable

Communication (including interrupts) is managed by the cache crossbar in OpenSPARC T1 (Fig. 3.1). Both out-bound (from the core-under-test to other resources) and in-bound (from other resources to the core-under-test) communications are disabled during CASP operation. All packets issued from the core-under-test are ignored by the crossbar. Packets intended for delivery to the core-under-test are buffered. The original OpenSPARC T1 design implements queues to buffer 2 packets per source for a core, and acknowledges the source once packets are dispatched from the queue. During CASP operation, the crossbar is modified to delay the dispatch of packets and the acknowledgement is sent to the source after the destination core resumes execution. Although queues with two entries are sufficient for normal operation, application characteristics must be considered to determine queue lengths for CASP (e.g. for frequent I/Os). Other ways to address this problem are: 1. Rely on internal buffers in most I/O device controllers; 2. Schedule tests in idle cores or during infrequent I/Os; 3. Retransmit undelivered packets or handle lost packets at the application-level; and, 4. Migrate tasks from core-under-test to a spare core.

Critical State Save and Restore

Since scan test is destructive, critical states of the core-under-test are saved before the scan test starts, and restored when the scan test completes. Critical states include program counters, thread states, processor states and control registers. Shadowed flip-flops (similar to data retention latches in ultra-low-power designs [Zyuban 02]) are used to save critical states. Once the scan test completes, values retained in shadowed flip-flops are reloaded into the processor core to resume operation. A total of 12,526 flip-flops are shadowed for each processor core in OpenSPARC T1. The shadow portions of these flip-flops can be designed to consume low power and operate at a very slow speed, and can be turned off

during normal operation. For designs with already existing shadowed flip-flops for data retention or test purposes [Kuppuswamy 04, Zyuban 02], the incremental cost of saving critical states is very small. For designs without such special flip-flops, the incremental area impact of using such special flip-flops is approximately 4% at the processor core level (after placement and routing). The chip-level area impact is even lower. The CASP test controller disables writes into the SRAM structures such as the register file, store buffer, and the trap stack array. Hence, their contents do not get overwritten during test, and the states of these components need not be saved.

Design and Validation Effort

The Verilog RTL of OpenSPARC T1 was modified to incorporate all the features above. We added or modified approximately 8,000 lines of Verilog code out of hundreds of thousands of lines in the original design. Most of the modification was straightforward (e.g., involving critical state saving) and did not require major changes to the normal operation of the design, which simplified the validation task. We verified the functionality of CASP by arbitrarily selecting a core for self-test during regression verification test runs, applying CASP tests to the selected core while regression testing continues (i.e., other cores continue execution while the selected core is isolated and tested), resuming normal operation of the selected core, and matching final results after regression tests complete.

3.3. CASP Test Data Storage

CASP test data (test patterns, expected responses, mask bits and control bits) is stored in non-volatile storage such as hard disks or FLASH memory. This approach allows test patterns to be pre-generated and shipped along with the system. The patterns may also be updated using patches after a system has been shipped. For scan-based CASP, test patterns, expected responses, and mask bits dominate test data volume. The required non-volatile storage (in bits) can be estimated as 3 × Number of test patterns × Number of flip-flops / Amount of test compression, where the factor of 3 accounts for test patterns, expected responses, and masks.

For OpenSPARC T1, there are 20,334 flip-flops per processor core, excluding the SRAM components such as the L1 Cache and TLB, and the register files. We organized these flip-flops into 10 scan chains. Table 3.1 reports stuck-at, transition, and Cadence True-Time delay test pattern counts and coverage values for an OpenSPARC T1 processor core. The stuck-at and transition patterns are generated using Synopsys TetraMax [Synopsys 07]. The True-Time delay patterns are generated using Cadence Encounter Test [Cadence 04]. We pessimistically invoked both tools with minimum test compaction effort, and did not assume any on-chip test compression. Since all processor cores are identical, separate test data storage is not required for each core. Hence, the required non-volatile storage is approximately 3 x 20,344 flip-flops x 7,193 patterns ≈ 52.3MBytes with no compression, and 5.3MBytes with modest 10X compression. Depending on the design, a higher compression ratio (>100X) may be achieved. This required amount of non-volatile storage is practical, since hard disks of PCs already have capacities of over 300 GBytes, the cost of FLASH memory is decreasing rapidly, and new high-density non-volatile memory technologies are emerging. These trends favor CASP considerably.

Table 3.1. Scan test patterns for an OpenSPARC T1 processor core (minimum test compaction effort).

	Pattern count	Test coverage
Stuck-at	609	99.49%
Transition	1,206	95.96%
True-Time	5,738	93.55%
Aggregate	7,193	N.A.

3.4. CASP Test Time and Test Data Transfer

Test time trade-offs for CASP are very different from production testing especially in the context of periodic testing for circuit failure prediction. That is why, as demonstrated in this section, it is practical to access test data from non-volatile memory such as hard disks or FLASH for CASP.

In our implementation, we have an on-chip buffer to store one test pattern and the corresponding response and mask, resulting in a buffer size of 3 × 20,334 flip flops x 1 pattern ≈ 7.5 KBytes (without compression). This buffer is shared among all processor cores. It is also practical to store compressed test data since most designs use on-chip test compression for test cost reduction. The required buffer size is then 768 Bytes and 154 Bytes for 10X and 50X compression, respectively.

Test pattern transfer time for hard disks is approximated by: Average seek-time + Size of data / Disk data transfer rate. We consider a worst-case scenario where the seek-time is included for every test pattern access. For FLASH memory, the read access time can be directly obtained from the transfer rate. Table 3.2 summarizes total transfer times, including seek times and data transfer rate (data transfer rate accounts for all controller and bus transfer overhead), for a single test pattern (7.5KBytes) for representative hard disks and FLASH storage.

Table 3.2. Access times for hard disks and FLASH memory and transfer times for a single test pattern (7.5KBytes uncompressed) [Gray 07].

Non-volatile storage type	Average seek time	Data transfer rate	Total transfer time
SCSI 15k rpm	3.5ms	75 MBps	3.6ms
SATA 10k rpm	4.6ms	60 MBps	4.722ms
FLASH	-	53 MBps	~0.138 ms

Test application time is dominated by scan shifts along the longest scan chain. The total number of scan shift cycles is approximately (Number of consecutive test patterns applied + 1) × Length of longest scan chain. In the worst-case scenario, a single test pattern will be accessed from non-volatile memory and applied, and the corresponding response will be compared against the expected response during the shift-out operation. In this case, for OpenSPARC T1, since the longest scan chain contains 2,034 flip-flops, the total number of scan shift cycles is 2 x 2,034 flip-flops x 7,193 patterns. The overall data transfer time is the transfer time for 1 pattern x 7,193 patterns. Table 3.3 shows the worst-case overall CASP test time for all test patterns, including all test time components, for OpenSPARC T1 with a 150 MHz test clock.

Even such worst-case test times are very practical, especially for circuit failure prediction. As Table 3.3 indicates, frequent CASP tests, e.g., once every few minutes with FLASH, are possible. In addition, there are several ways to reduce CASP test time:

1. FLASH significantly reduces transfer time. Systems with hard disks replaced by FLASH are shipped today [Dell 07].

2. Pre-fetching test patterns into on-chip buffers can hide test pattern read access and transfer times.

3. Task migration from the core-under-test to a spare core enables continuous CASP test. Experiments on an ARM multi-core processor indicate a 0.5ms OS switch overhead, which is very short from system perspective [Inoue 07].

Table 3.3. Worst-case overall CASP test time for an OpenSPARC T1 processor core.

Non-volatile storage type	Total transfer time	Test application time	Overall test time
SCSI 15k rpm	~26sec	~0.2s	~26.2sec
SATA 10k rpm	~34sec	~0.2s	~34.2sec
FLASH	~1sec	~0.2s	~1.2sec

4. Related Work

Previous work on on-line self-test can be classified into the following categories: 1. Built-In Self Test (BIST) [Bardell 87, Chen 03, Constantinides 06, Gupta 96, Krstic 02, Kusko 01, McCluskey 84, 86, Parvathala 02, Shen 98, Touba 96]; 2. Roving emulation [Breuer 86]; and, 3. Periodic functional testing [Corno 96, Karri 98, Krantis 02, 06, Paschalis 05, Weglarz 04]. Table 4.1 presents a comparative analysis.

Table 4.1. Comparison of on-line test techniques.

	CASP	Logic BIST	Roving emulation	Periodic functional test
Coverage	Very high	Low	Low	Low
Area cost	Low	High for high coverage	Moderate	Moderate
Test data storage	Large but low-cost	Small	Small	Small
Test time impact	Short for system, very little with spare core	Targets test floor	Short with spare core	Short for low coverage
Design effort	Moderate	High	High	Moderate
Flexibility	High	Low	–	Low

A major advantage of CASP is the high test coverage for both datapath and control parts of a design. Special techniques for high test coverage with pseudo-random logic BIST can be expensive [Touba 96, Wunderlich 96]. While BIST can apply high-coverage patterns for datapath circuits, control portions often have coverage problems. Pseudo-exhaustive BIST suffering from long test lengths and logic modifications can be intrusive. Functional BIST and periodic functional test techniques, while useful, have coverage problems especially for the control logic of a design. Coverage of roving emulation depends on applications executing during the checking window and cannot be guaranteed *a priori*. CASP avoids these problems by storing high-quality test patterns. Proliferation of low-cost, high-density non-volatile storage and test compression favor CASP. CASP is flexible because test patterns can be modified after a system is deployed in the field.

CASP test time may seem high compared to other techniques. However, as demonstrated in Sec. 3.4, the overall system-level performance impact is extremely small especially for circuit failure prediction. Moreover, test data read access and transfer time, which account for a significant part of the overall test time, can be hidden through pre-fetching. The test time impact can also be minimized through clever test scheduling (e.g., idle cores) or by task migration from the core-under-test to a spare core.

As detailed in Sec. 3, CASP design and validation efforts are moderate. From a testing perspective, the amount of DFT effort is the same as that for test compression in a processor core. Design efforts for CASP architectural features and the CASP test controller are moderate. The effort can be even lower for checkpointed processors. In comparison, Logic BIST design efforts can be very high especially for X-bounding and delay testing. The design effort associated with roving emulation can be very high because duplication-based checking of processor cores can be complex (as explained in [Bernick 05]).

5. Conclusions and Future Work

CASP is a concurrent and autonomous chip self-test technique which enables design of robust systems to overcome major reliability challenges in scaled CMOS technologies. CASP avoids major limitations of existing on-line testing techniques while enabling highly thorough testing at very low cost. This is because CASP utilizes major technology trends, such as proliferation of multi-core architectures, availability of high density and low-cost non-volatile memory, and use of test compression, to its benefit. CASP is effective for circuit failure prediction, self-test-based error detection and self-diagnosis. The CASP implementation in the OpenSPARC T1 multi-core processor demonstrates its practicality.

Future CASP research directions include: 1. Efficient CASP test scheduling to minimize system-level performance and power impact; 2. Integration of low-power testing techniques [Bonhomme 01, Chandra 02, Saxena 01, Whetsel 00] into CASP; 3. Task migration from core-under-test to a spare core to minimize performance impact; 4. Operating system and virtualization support for core isolation; 5. CASP for on-line functional testing and on-line Schmoos (the latter may require separate power supplies for processor cores); and, 6. CASP for production test and post-Silicon debug. It is important to analyze the vulnerability of CASP to side channel attacks [Yang 04]. We expect that decompressors and compactors for test compression, e.g., [Mitra 04, Touba 06], will help avoid such vulnerabilities.

Acknowledgment

This research was supported in part by the FCRP Gigascale Systems Research Center (GSRC) and the National Science Foundation CAREER Award.

References

[Agarwal 07] Agarwal M., B. Paul, M. Zhang, and S. Mitra, "Circuit Failure Prediction and Its Application to Transistor Aging," *IEEE VLSI Test Symp.*, pp. 277-286, 2007.

[Al-Asaad 98] Al-Asaad, H., B.T. Murray, and J.P. Hayes, "Online BIST for Embedded Systems," *IEEE Design & Test of Computers,* pp.17-24, 1998.

[AMD 07] "AMD Multicore Technology Evolution," http://multicore.amd.com/us-en/AMD-Multi-Core/Technology-Evolution.aspx.

[Azul 07] "Azul Systems First to Deliver 48-Way Multicore Chip, Redefining Standard in Enterprise Computing," http://www.azulsystems.com/press/032706_vega2.htm

[Bardell 87] Bardell, P.H., W.H. McAnney, and J. Savir, *Buit-In Test for VLSI: Pseudorandom Techniques*, Wiley, 1987.

[Bardell 91] Bardell P.H., and M.J. Papointe, "Production Experience with Built-In Self-Test," *Proc. Intl. Test Conf.*, pp. 28-36, 1991.

[Bernick 05] Bernick, D., *et al.*, "Non-Stop Advanced Architecture," *Intl. Conf. Dependable Systems and Networks*, pp. 12-21, 2005.

[Bonhomme 01] Bonhomme, Y., *et al.,* "A Gated Clock Scheme for Low Power Scan Testing of Logic ICs or Embedded Cores," *Proc. Asian Test Symp.,* pp. 253-258, 2001.

[Borkar 05] Borkar, S., "Designing Reliable Systems from Unreliable Components: The Challenges of Transistor Variability and Degradation," *IEEE Micro*, Vol. 25, Issue 6, pp. 10-16, 2005.

[Borkar 07] Borkar, S., "Thousand Core Chips – A Technology Perspective," *Proc. Design Automation Conf.*, pp. 746-749, 2007.

[Breuer 86] Breuer, M., and A. Ismaeel, "Roving Emulation as a Fault Detection Mechanism," *IEEE Trans. Comput.,* pp. 933-939, 1986.

[Cadence 04] "Delay Test Methods in Encounter Test," http://sourcelink.cadence.com/docs/files/Application_Notes/2004/EncounterTest_delay_test.pdf.

[Chandra 02] Chandra, A., and K., Chakrabarty, "Low-Power Scan Testing and Test Data Compresiion for System-on-a-Chip," *IEEE Trans. CAD,* pp. 597-604, 2002.

[Chen 03] Chen, L., *et al.*, "A Scalable Software-based Self-test Methodology for Programmable Processors," *Proc. Design Automation Conf.*, pp. 548-553, 2003.

[Cisco 07] http://newsroom.cisco.com/dlls/partners/news/2004/pr_prod_06-09.html

[Constantinides 06] Constantinides, K., *et al.*, "BulletProof: A Defect-Tolerant CMP Switch Architecture," *Proc. Intl. Symp. High-Performance Computer Architecture*, 2006.

[Corno 96] Corno, M., *et al.*, "On-line Testing of an Off-the-shelf Microprocessor Board for Safety-critical Applications," *European Dependable Computing Conf.,* pp. 190-202, 1996.

[Dell 07] http://www.dell.com/content/topics/global.aspx/corp/pressoffice/en/2007/2007_04_24_rr_000?c=us&l=en&s=corp.

[Gray 07] Gray, J., and B. Fitzgerald, "FLASH Disk Opportunity for Sever Applications," http://research.microsoft.com/~Gray/papers/FlashDiskPublic.doc.

[Gupta 96] Gupta, S.K., and D. Pradhan, "Utilization of On-Line (Concurrent) Checkers during Built-In Self-Test and Vice Versa," *IEEE Trans. Computers,* pp. 63-73, 1996.

[IBM 07] "POWER processor-based blade servers," http://www-03.ibm.com/systems/bladecenter/power-based.html

[Inoue 07] Inoue, H., *et al.*, "Dynamic Security Domain Scaling on Symmetric Multiprocessors for Future High-End Embedded Systems," *Proceedings of Intl. Conf. on CODES & ISSS,* pp. 39-44, 2007.

[Intel 07] "Intel Multicore," http://www.intel.com/multi-core.

[Iyengar 06] Iyengar, V., *et al.*, "At-Speed Structural Test for High-Performance ASICs," *Proc. Intl. Test Conf.,* pp. 1-10, 2006.

[Karri 98] Karri, R., and N. Mukherjee, "VBIST: An Integrated approach to On-Line/Off-Line BIST," *Proc. Intl. Test Conf.*, pp. 910-917, 1998.

[Kuppuswamy 04] Kuppuswamy, R., *et al.*, "Full Hold-Scan Systems in Microprocessors: Cost/Benefit. Analysis," *Intel Technology Journal*, Vol. 18, No. 1, Feb. 2004.

[Kusko 01] Kusko, M., *et al.*, "99% AC Test Coverage Using Only LBIST on the 1-GHz IBM S/390 Z-series 900 Microprocessor," *Proc. Intl. Test Conf.,* pp. 586-592, 2001.

[Krantis 02] Krantis, N., *et al.*, "Effective Software Self-Test Methodology for Processor Cores," *Proc. DATE*, 2002.

[Krantis 06] Krantis, N., *et al.*, "Optimal Periodic Testing of Intermittent Faults in Embedded Pipelined Processor Applications," *Proc. DATE,* pp. 65-70, 2006.

[Krstic 02] Krstic, A., *et al.*, "Embedded Software-Based Self-Test for Programmable Core-Based Designs," *IEEE Design & Test of Computers*, pp. 18-27, 2002.

[McCluskey 84] McCluskey, E.J., "Verification Testing – A Pseudoexhaustive Test Technique," *IEEE Trans. Computers*, pp. 541-546, June 1984.

[McCluskey 86] McCluskey, E.J., *Logic Design Principles,* Prentice Hall, 1986.

[McNairy 04] McNairy, C., and R. Bhatia, "Montecito: The Next Product in the Itanium Processor Family", *Hot Chips 16,* 2004.

[Mitra 04] Mitra, S., and K.S. Kim, "X-Compact: an efficient response compaction technique," *IEEE Trans. CAD,* pp. 421-432, 2004.

[Molyneaux 07] Molyneaux, R., *et al.*, "Design-for-Testability Features of the Sun Microsystems Niagara2 CMP/CMT SPARC Chip", *Intl. Test Conf,* 2007.

[nVidia 07] "GeForce 7950 GPUs," http://www.nvidia.com/page/geforce_7950.html.

[Parvathala 02] Parvathala, P., K. Maneparambil, and W. Lindsay, "FRITS: A Microprocessor Functional BIST Method," *Proc. Intl. Test Conf.*, pp. 590-598, 2002.

[Paschalis 05] Paschalis, A., and D. Gizopoulos, "Effective Software-Based Self-Test Strategies for On-Line Periodic Testing of Embedded Processors," *IEEE Trans. CAD,* pp. 88-99, Jan. 2005.

[Saxena 01] Saxena, J., *et al.*, "An Analysis of Power Reduction Techniques in Scan Testing," *Intl. Test Conf.,* p. 670-677, 2001.

[Shen 98] Shen, J., and J.A. Abraham, "Native mode functional test generation for processors with applications to self test and design validation," *Proc. Intl. Test Conf.,* pp. 18-23, 1998.

[Sun 06] "OpenSPARC T1 Microarchitecture Specification," http://opensparc-t1.sunsource.net/specs/OpenSPARCT1_Micro_Arch. pdf.

[SUN 07] "UltraSPARC T2 Processor," www.sun.com/processors/UltraSPARC-T2/datasheet.pdf.

[Synopsys 07] TetraMAX ATPG User Guide, version Z-2007.03.

[Touba 06] Touba, N. A., "Survey of Test Vector Compression Techniques," *IEEE Design & Test of Computers,* pp. 294-303, 2006.

[Touba 96] Touba, N. A., and E. J. McCluskey, "Altering a Pseudo-Random Bit Sequence for Scan-Based BIST," *Proc. Intl. Test Conf.*, pp. 167-175, 1996.

[Weglarz 04] Weglarz, E., K. Saluja, and T.M. Mak, "Testing of Hard Faults in Simultaneous Multithreaded Processors," *Proc. Intl. Symp. On-line Testing*, pp. 95-100, 2004.

[Wunderlich 96] Wunderlich, H.J., and G. Kiefer, "Bit-Flipping BIST," *Proc. ICCAD*, pp. 337-343, 1996.

[Whetsel 00] Whetsel, L., "Adapting Scan Architectures for Low Power Operation," *Proc. Intl. Test Conf.,* pp. 863-872, 2000.

[Yang 04] Yang, B., K. Wu, and R. Karri, "Scan Based Side Channel Attack on Dedicated Hardware Implementations of Data Encryption Standard," *Proc. Intl. Test Conf.*, pp. 339-344, 2004.

[Zyuban 02] Zyuban, V., and S. Kosonocky, "Low Power Integrated Scan Retention Mechanism," *Intl. Symp. Low Power Design*, pp. 98-102, 2002.