Fault Tolerance in Multicore Processors With Reconfigurable Hardware Unit

Rajesh S, Vinoth Kumar C, Srivatsan R, Harini S and A.P. Shanthi
Department of Computer Science & Engineering,
College of Engineering Guindy, Anna University, Chennai, India

Abstract - The present day systems are in need of high level of fault tolerance in the Multicore processors without substantial loss of overall performance. The commodity processors that are available now have handled mainly soft errors (transient errors) and a very small amount of work is done for handling hard faults. In this paper we propose to include a Reconfigurable Hardware Unit (RHU) inside the core which can detect and isolate the faults in the functional units inside a core using stored test patterns. Once the faulty unit is isolated, a part of the RHU is reconfigured by loading stored configuration to perform the functions of the faulty unit and the register values of the faulty unit is forwarded to the reconfigured RHU. The test patterns and configurations should be stored in fast access non-volatile storage devices such as flash memory. This improved architecture will help to solve many fault tolerance issues with no visible loss of performance at low cost and space.

Key Words - Multicore, CMP, Reconfigurable Hardware Unit, Test Patterns.

1. Introduction

The number of cores in a die is increasing at a high rate; subsequently, the functional units are shrinking,making it more susceptible to hardware errors. Permanent or intermittent hardware faults, caused by defects in the silicon or metallization of process package and wear out over time, lead to “hard faults”. Transient faults (or “soft errors”), which cause random bit values to change erroneously, may be caused by electrical noise (e.g., crosstalk) or external radiation (e.g., alpha radiation from impurities).

At present most of the commodity CMPs have handled many of the soft errors that can occur inside a core. But handling hard faults is a tedious task. Replication of each of the functional units is not cost and space friendly. Even if we replicate, the complete utilization of all the functional units is not possible. The improvements in the field of Reconfigurable Computing could help us solve the fault tolerance issues.

Reconfigurable Computing is getting more and more important in the (embedded) computing world. Field Programmable Gate Arrays (FPGAs), Field Programmable Transistor Arrays (FPTAs) and Complex Programmable Logic Devices (CPLDs) are used as building blocks for reconfigurable computing. Designing architectures for (embedded) computer systems using reconfigurable hardware is becoming more popular, now that classical drawbacks are diminishing.[1]

This is mostly because of the speed/flexibility trade-off which holds in architectural design.

[1] Student Author:Rajesh S,Vinoth Kumar C,Srivatsan R,Harini S.

With the use of reconfigurable hardware, one tries to fill the gap between a hardware-only (Application Specific Integrated Circuit-ASIC) and a software-only (General Purpose Processors-GPP) solution [8]. It is intended to achieve a higher performance than a software-only solution, and maintain more flexibility than a hardware-only solution.

The effectiveness of reconfigurable computing in a general purpose, high performance processor as in a Multicore processor has not yet been evaluated. We propose to integrate a reconfigurable hardware unit inside a core. The RHU, without disturbing the normal functionalities of the core, tests the functional units for faults which can take place in a round-robin or event driven manner, and identifies the exact location of the fault. When a fault is identified, the RHU is reconfigured to take over the functions of the faulty unit. Thus, tolerance in many of the functional units inside a core can be achieved.

2. Related Work

Contemporary processors like the UltraSPARC T1, which are manufactured on cutting-edge process technology, are especially prone to soft errors. With this problem in mind, Sun systematically designed the UltraSPARC T1 processor with the appropriate level of protection of its on-chip memories. In general, the UltraSPARC T1 processor protects memory arrays with either Single Error Correction / Double Error Detection (SEC/DED) or parity protection. Redundant arrays are protected with parity, while non-redundant arrays are protected with ECC (Error Correcting Code). Table 1 lists the UltraSPARC T1 processor's on-chip memories and its corresponding protection mechanism.

TABLE 1 :On -Chip Memory Protection
Memory Array Protection

Integer Register File	ECC
Floating Point Register File	ECC
L1 Instruction Cache-Data	Parity/retry
L1 Instruction Cache – Tag	Parity/retry
Instruction TLB	Parity/retry
Data TLB	Parity/retry
L1 Data Cache – Data	Parity/retry
L1 Data Cache – Tag	Parity/retry
L2 Cache – Data	ECC
L2 Cache –Tag	ECC
L2 Cache Scrubber	Yes

The UltraSPARC T1 processor's Chipkill mechanism1 can correct any error contained within a single memory nibble (4 bits), and detect any uncorrectable errors contained within any two nibbles. Another mechanism implemented in the UltraSPARC T1 processor to ensure main memory reliability is memory scrubbing. Each of the UltraSPARC T1 processor's four memory controllers has a background error scanner/scrubber to reduce the incidence of multi-nibble errors. At the International Reliability Physics Symposium1, Sun showed that implementing power and thermal management features can dramatically increase both the lifetime and reliability of the device by up to 24 times while maintaining or improving device performance.

Li et al. [1] have proposed a method (CASP) for autonomous testing of cores in Multicore environment by adding a hardware unit. CASP 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.

ARGUS [3] exploits the fact that the core performs only four basic operations, choosing the sequence of instructions to execute (control flow), performing the computation specified by each instruction, passing the result of each instruction to its data-dependent instructions (data flow), and interacting with memory. By checking all these activities most of the errors that can occur inside a core can be detected.

Bower et al. [4] have proposed a DIVA checker that detects an error in an instruction and increments a small saturating error counter for every field deconfigurable unit (FDU) used by that instruction, including the DIVA checker. A hard fault in an FDU quickly leads to an above-threshold error counter for that FDU and thus diagnoses the fault.

Bell et al. [6] have proposed that redundant execution on chip multiprocessors helps in detecting the faults with no major impact on performance. Errors can be detected by buffering retired stores in a store comparator queue where they are compared to identical stores executed on a second thread. If a mismatch is detected in either a store's data or address, an error is signaled to the processor so that it can respond appropriately.

3. Existing Problems

The present transient fault detection is limited to storage arrays such as register files, cache and memory arrays. It is implemented now using certain registers which maintains the parity information which can be used to detect the errors.

None of the current generation CMPs can tolerate errors in the associated cache circuitry or interconnect. . For example, if all L2 cache banks are shared, and addresses are interleaved among the banks, a transient failure in the cache controller state machine could lead to erroneous setting of a coherence bit. Note that ECC on the coherence state bit would not prevent this error because the fault is in the cache controller logic and not the actual coherence bit. Such an error could affect an entire socket.

There is no fault isolation in Opteron, Xeon and Niagara; an error originating in any core can propagate to all the other cores through the shared system components. For example, if the L2 cache is shared among various cores, then any error in the cache controller will affect all the cores that share these caches. Thus, any fault in shared resources is difficult to isolate.

All the architectures have sophisticated techniques like chip kill, background scrubbing, and DIMM sparing to tolerate failures' here is no tolerance to failures in memory access control circuitry. A failure in any memory controller or anywhere in the interconnect would affect all the cores.

4. Proposed Solutions

The RHU which is integrated into the core has the following functions - Detection, Isolation and correction.

The Testing unit configured in the RHU can function as Signal Tracker or Control Flow or Data Flow or Computation Checker. Testing can be done in event-driven and round-robin fashion.

The signals that are generated in the core are monitored for a while and if any abnormal pattern of signals is identified, a tester is configured in the RHU which tests the core for faults. When a particular unit is under testing, the RHU is configured to perform the functions of the unit under test. This helps us to test each functional unit without disturbing the normal operations of the core. When the testing is completed, the functions are again transferred to the original unit.

While testing, the RHU can be configured as a control flow checker that periodically verifies that the runtime execution path is valid with respect to the static control flow graph (CFG) of the program binary. If the static and dynamic CFGs conflict, an error is detected which implies the core has a faulty unit.
The faulty unit can be isolated by testing the core using stored test patterns in non-volatile storage. This requires a test controller and on-chip buffer for scheduling the self-test in the processor core. The basic idea is to store very thorough test patterns in non-volatile storage, such as hard disks or FLASH memory.

The test controller will
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.
4.Identify the faulty unit in the core.

The configuration of each of the functional units is stored in FLASH memory. When a faulty unit is isolated, the configuration of that unit is loaded into the RHU and this reconfigured unit takes over the functions of the faulty unit. This process helps the core to recover from serious faults with minimal delay.

The values of the registers of the faulty unit has to be transferred to the reconfigured unit .This can be done by including pseudo instructions which transfers the registers of faulty unit to RHU.

5. Proposed Implementation

The Reconfigurable Hardware Unit can be implemented using FPGA (Field Programmable Gate Array) or CPLD (Complex Programmable Logic Devices). FPGA is more flexible and efficient compared to CPLD.

Xilinx Virtex is a commercially available FPGA device. The code of a single core of OpenSPARC can be downloaded into Xilinx Virtex and a flash memory should be attached. The flash memory should be in such a way it can also be shared among other cores when used in a Multicore environment.

Sun's OpenSparcT1 architecture is used for the implementation of the proposed work as it provides various open source tool's support to simulate the design.A High level VHDL model for the testing, isolation and correction units is generated. The logic is partitioned. Each part was re-described in a lower level description (RTL) required for the circuit synthesis, optimization and mapping to the specific technology by assigning current FPGA family and device. The resulting optimized circuit description was verified through extensive simulation after which the layout was created (Layout synthesis) and finally, on chip verification was executed by using C++ programming to connect PCI bus to the design ports and to test the design.

5.1 Architecture Diagram

Architectural features of the proposed work include
1.Support for stalling and draining the pipeline,invalidating the cache,2.Support for restoring states and enabling communication.These architectural features can be introduced to OpenSparc T1 with moderate design effort as they are supported by the OpenSparcT1.
Sun's OpenSparc Architecture have eight cores and main functional units of a core are shown in the Fig.2.We propose to include a RHU unit into each of the cores of the OpenSPARC and a flash memory which can be shared by all the cores. (The included units are shown by dotted circle in a core in Fig.2.). The RHU should be able to meet the space constraints on the core. The flash memory is used to store test patterns and configurations so that the reconfiguration latency is minimum.

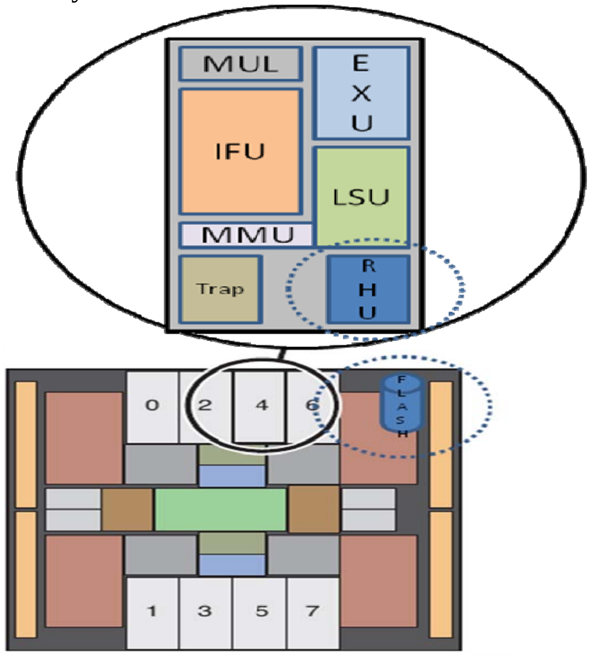

Fig.2. Proposed Architecture

In future the faults in RHU should also be identified by self-test mechanism and the identified faulty component must not be used for later purposes. We should also handle error propagation between cores due to faults in shared resources.

6. Conclusion & Future Work

As the size of the functional units decreases the number of faults that can occur in a core increases drastically. This makes the need for fault tolerant systems more important. We have proposed to utilize the power and flexibility of RHU to bring in fault tolerance in the core at unit level. Our system tests the core at unit level and replaces the faulty component with minimal reconfiguration latency. Thus, the reliability of the processors increases in exchange to some additional cost and space inside the core.

References:

[1] [Yanjing 08] Yanjing Li,Samy Makar,Subhasish Mitra ,"CASP: Concurrent Autonomous Chip Self-Test Using Stored Test Patterns", Design Automation and Test in Europe ,2008.

[2] [Sumit 06] Sumit Dharampal Mediratta,Jeffrey Draper,"Achieving On-chip Fault-tolerance Utilizing BIST Resources ",Proceedings of the 5th WSEAS International Conference on Circuits, Systems, Electronics, Control & Signal Processing ,2006.

[3] [Meixner 07] Albert Meixner Michael E. Bauer Daniel J. Sorin, "ARGUS: Low-cost,comprehensive error detection in simple cores",IEEE,2007.

[4] [Bower 05] Fred A. Bower1,3, Daniel J. Sorin2, and Sule Ozev2,"A Mechanism for Online Diagnosis of Hard Faults in Microprocessors", International Symposium on Microarchitecture ,2005.

[5] [Osamah 07] Dr. Osamah A. Rawashdeh,"A Reconfigurable Architecture for Fault-Tolerant Distributed Embedded Systems", EURASIP Journal on Embedded Systems,2007.

[6] [Bell 07] Gordon B. Bell, Mikko H. Lipasti, "Achieving Fault Detection and Performance on CMPs", ACM Computing Surveys (CSUR) ,2007.

[7] [Compton 02] K.Compton and S.Hauck , "Reconfigurable Computing: A survey ofsystems and software", ACM Computing Surveys (CSUR),2002.

[8] [Sellers 68] F. F. Sellers et al. Error Detecting Logic for Digital Computers. McGraw Hill Book company, ACM Computing Surveys (CSUR),1968.

[9] [Aggarwal 07] Nidhi Aggarwal, Parthasarathy Ranganathan, Norman P. Jouppi,James E. Smith, Kewal K. Saluja, and George Krejci,"Motivating Commodity Multi-Core Processor Design for System-level Error Protection",2007.

[10] [Aggarwal07]Nidhi Aggarwal ,Parthasarathy Ranganathan,James E. Smith," Configurable Isolation: Building High Availability Systems with Commodity Multi-Core Processors",2007.