INTEGRATION, the VLSI journal 55 (2016) 183–193

Contents lists available at ScienceDirect

INTEGRATION, the VLSI journal

journal homepage: www.elsevier.com/locate/vlsi

Stochastic testing of processing cores in a many-core architecture

Arezoo Kamran*, Zainalabedin Navabi

School of Electrical and Computer Engineering, University College of Engineering, University of Tehran, Tehran, Iran

ARTICLE INFO

Article history:
Received 25 August 2015
Received in revised form
2 April 2016
Accepted 15 June 2016
Available online 16 June 2016

Keywords:
Many-core
Online test
Peti Net modeling
Reliability
Software-based self-test

ABSTRACT

A promising solution to reliability challenges in nano-scale fabrication technologies is self-test and reconfiguration. In this direction, we propose an autonomous test mechanism for online detection of permanent faults in many-core processors. Several hardware test components are incorporated in the many-core architecture. Some of these components distribute software-based self-test routines among the processing cores and make each test routine accessible for a limited amount of time. A processing core that has an idle slot executes the test routine, otherwise it skips it without loss of test continuity. Several components of the proposed test architecture monitor behavior of the processing cores during execution of test routines, detect faulty cores, and make their omission from the system possible. We propose the use of an extended form of *Petri NET* modeling method to model and analyze the proposed test mechanism and tune our test architecture to preserve quality of test, and at the same time, manage the overall test time. Our experimental results show that test time and hardware overhead of the proposed test mechanism are low and its performance overhead is zero. Furthermore, the proposed test architecture can efficiently scale to a many-core with a large number of processing cores.

1. Introduction

Nano-scale fabrication technologies have provided the capability of producing single-chip many-core processors with up to hundreds of processing cores. These many-core architectures have opened new horizons in microprocessor industry and are going to be a dominant trend in general purpose microprocessors [1–5], and also in specialized processing units such as Graphics Processing Units (GPUs) and Network Processing Units [6,7]. However, extensive miniaturization and large scale integration in new fabrication technologies have led to unpleasant side effects such as production yield drop, infant mortality, and accelerated aging [8–10] in all nano-scale semiconductor devices including many-core processors. In fact, because of non-aggressive burn-in testing, more pronounced aging effects, and incomplete testing and verification processes due to increased time-to-market pressure in new fabrication technologies, systems fabricated in these technologies may experience faults (including early defects or latent faults) and fail any time in the field. Therefore, a one-time factory testing is not sufficient in future VLSI components. In fact, success of new fabrication technologies heavily depends on incorporating test architectures and mechanisms into the design to deal with low yield and frequent in-field failure of components, and guarantee long-life reliable operation of future chips.

To enhance reliability of many-core processors, traditional redundancy based techniques can be used to utilize inherent redundancy that already exists in processing cores. But redundancy-based techniques devote a considerable portion of additional devices provided by miniaturized fabrication technologies to protect other on-chip components, and have no contribution for improving the performance [11]. A promising solution to the reliability challenges of new fabrication technologies is self-test and self-reconfiguration with no or limited external control [12]. Various researches has been conducted in this area. Austin et al. [13], used dynamic verification concepts and introduced a micro-architectural-based technique called DIVA (Dynamic Implementation Verification Architecture) enabling a complex processor to dynamically detect functional and electrical faults. Constantinides et al., [14] have added several new instructions to the instruction set architecture of a processor. The instructions are referred to as Access-Control Extensions (ACE) that provide access and control of the processor internal state via software. A special firmware periodically suspends the processor normal operation, stores the processor state, and applies the ACE instruction to the processor core to check its health status. Several works such as [15–17] have used predetermined scan-based test patterns to test processing cores in a Chip Multi-Processor (CMP). They have added various hardware components to apply a test to the processing cores and check their responses. Several works [18–20] have used natural core-level redundancy available in a *CMP* and used processing power of a processing core to test correct operation of another processing core. [21,22] have incorporated *BIST* (Built-In

* Corresponding author.
E-mail addresses: akamran@ut.ac.ir (A. Kamran), navabi@ut.ac.ir (Z. Navabi).

http://dx.doi.org/10.1016/j.vlsi.2016.06.005
0167-9260/© 2016 Elsevier B.V. All rights reserved.

Self-Test) components into the *CMP* architecture and have provided a mechanism that periodically triggers the *BIST* circuitry to test various components of a *CMP*.

An important processor testing approach is Software-Based Self-Testing (*SBST*) that tests a processor by its native instructions. *SBST* techniques are non-intrusive in nature and facilitate at-speed testing that is very important in new fabrication technologies. Additionally, *SBST* techniques remove the over-testing problem associated with nonfunctional testing techniques and can be reused for online periodic testing of processors. Many research groups have worked on *SBST* of single core processors [23–29]. Today *SBST* is a mature technique for testing simple microprocessors and is an integral part of the manufacturing test process for single core processors [27].

Thus far, *SBST* techniques have mainly focused on unicore processors. An emerging trend in this research area is adapting these techniques to multicore and many-core processors [9,30,31]. In [30] a software-based self-test approach has been developed for symmetric shared-memory multiprocessors considering the most common interconnection architectures, shared bus and crossbar switch. In a later work [31], a test program parallelization method has been proposed to accelerate online permanent fault detection in many-core architectures. However, execution of test routines in multicore architectures imposes severe contention on shared resources, including shared memory subsystem and interconnection network, limiting the applicability of currently proposed techniques for online testing of processor architectures with a large number of processing cores.

In this work, we propose a test mechanism for online detection of permanent faults in homogeneous many-core processors. In this test architecture, several test components are incorporated in the many-core architecture that autonomously and concurrent with the system normal operation, distribute software-based self-test routines among the processing cores, monitor behavior of the processing cores during execution of the test routines, detect faulty cores, and make their omission from the system possible.

In order to utilize short idle times of the processing cores, test data is segmented into small pieces, called *test snippet*s. Individual *test snippet*s are distributed among the processing cores and are made accessible for them for a limited period of time. If a processing core has an idle slot during a period that a *test snippet* is available, it executes the *test snippet*, otherwise it skips execution of that portion of test. Our test mechanism is designed in a way that it supports skipping of *test snippet*s at the expense of losing test quality, but without any effect on the integrity of the whole test mechanism.

In order to make analysis of this stochastic system possible, and as an important contribution of this work, we have modeled the proposed test mechanism using an extended form of *Stochastic Petri Net* (*SPN*). We have analyzed the *Petri Net* model of the system, and used the results to decide on the system parameters to preserve quality of test at a desired level of confidence, with a minimum possible test time. Furthermore the results can be used to tune the test mechanism for less test time, or less hardware overhead. Test time, hardware overhead, and scalability of the proposed test mechanism are important parameters that have been investigated in this work.

The rest of this paper is organized as follows: Section 2 presents details of the proposed test mechanism. Section 3 discusses minimization of test time. Section 4 briefly introduces *Petri NET* modeling basics, and then shows how we can use this method to model and analyze the proposed test mechanism. Section 5 investigates experimental results, and finally Section 6 concludes the paper.

2. Proposed test mechanism

In this section, we introduce architectural details of a test mechanism for online fault detection of processing cores of a homogeneous many-core processor architecture. We assume that the many-core processor is composed of several identical nodes, each of which consists of a processing core, several cache blocks, and some hardware facilities for communications with other nodes. Test distribution in our proposed test architecture is independent of core communications, and utilizes a special test distribution logic that is dedicated to this purpose only. Therefore, our proposed test architecture does not make any assumption about the communication infrastructure of the many-core processor.

Even though, the proposed test mechanism provides an easily extendable platform for testing various components of a many-core, including cores, routers, and memories, this paper mainly focuses on testing the processing cores. In this work, processing cores are being tested using a software-based self-test approach. *SBST* methods are non-intrusive and provide the capability of at-speed testing of the processing cores that is crucial in continuously shrinking fabrication technologies with more prominent delay defects.

Instead of a pure *SBST* technique, in our proposed *SBST* technique, small amount of assisting hardware is incorporated in the many-core architecture. These hardware test components distribute software-based self-test routines among the processing cores of the many-core processor, detect idle processing cores and switch them to test mode to execute the software-based test routines, and monitor the behavior of the processing cores during the test.

Fig. 1 shows hardware components added near a processing core. Several adjacent processing cores share these additional test components. A local test controller called *cluster tester* receives a test routine from a shared global controller called *Autonomous Chip Tester* (*ACT*), and stores it in a small local buffer (called *test-snippet buffer*). When this *cluster tester* is triggered by a proper command from *ACT*, it starts monitoring the activity of neighboring processing cores. When a processing core becomes idle, the *cluster tester* disconnects the idle processing core from the communication infrastructure and the memory subsystem, and

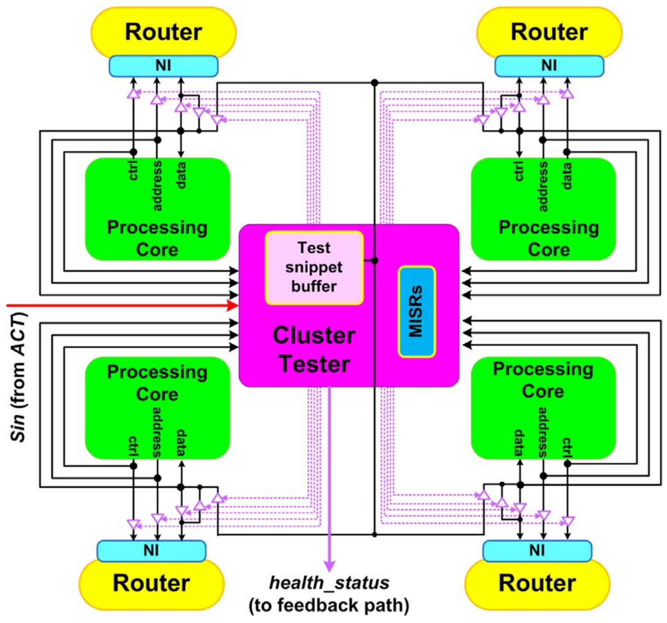

Fig. 1. Hardware test components situated near the processing cores.