

**EMPLOYING REDUNDANCY BASED
TECHNIQUES TO PROVIDE RELIABILITY,
SECURITY AND ACCOUNTABILITY IN
MODERN PROCESSORS**

RAJSHEKAR K



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY DELHI
SEPTEMBER 2017

©Indian Institute of Technology Delhi - 2017
All rights reserved.

**EMPLOYING REDUNDANCY BASED
TECHNIQUES TO PROVIDE RELIABILITY,
SECURITY AND ACCOUNTABILITY IN
MODERN PROCESSORS**

by

RAJSHEKAR K

Department of Computer Science and Engineering

Submitted

in fulfillment of the requirements of the degree of **Doctor of Philosophy**

to the



Indian Institute of Technology Delhi

SEPTEMBER 2017

Certificate

This is to certify that the thesis titled **EMPLOYING REDUNDANCY BASED TECHNIQUES TO PROVIDE RELIABILITY, SECURITY AND ACCOUNTABILITY IN MODERN PROCESSORS** being submitted by **Mr. RAJSHEKAR K** for the award of **Doctor of Philosophy in Computer Science and Engineering** is a record of bona fide work carried out by him under my guidance and supervision at the **Department of Computer Science and Engineering, Indian Institute of Technology Delhi**. The work presented in this thesis has not been submitted elsewhere, either in part or full, for the award of any other degree or diploma.

Smruti Ranjan Sarangi
Assistant Professor
Department of Computer Science and Engineering
Indian Institute of Technology Delhi
New Delhi- 110016

Acknowledgements

I firstly thank my advisor, Dr. Smruti Sarangi, for all his help and guidance. He made me believe five years ago that I could be a PhD scholar, and he has now made me believe that I can advise PhD scholars myself. Thank you Sir.

I thank all the faculty in the Architecture group – Prof. Balakrishnan, Prof. Anshul Kumar, Prof. Panda and Dr Kolin Paul – and Dr. Shouri Chatterjee from the Department of Electrical Engineering, for reviewing my work and providing valuable insights.

I thank all my friends here at IITD for all the technical discussions that aided my research and deepened my understanding, as well as for all the other experiences that made life enjoyable and thought-provoking.

I thank all the boys back home for always being there. It made all the difference.

I thank all the boys from BMSCE for their support and encouragement. I got into IITD because of you guys!

I thank my parents for all their support and understanding, and for the sacrifices they have made the last seven and a half years I have been away from home.

I thank God, for all and everything.

Rajshekar K

Abstract

Though the classical goals in processor design – performance, power and area – are still being actively pursued, various new areas such as reliability and security have come into prominence.

As the transistors have become smaller, the threat of soft errors has increased. Mission critical systems are forced to employ chips fabricated at technology nodes that are at least 2-4 generations older, thereby suffering from low performance [1].

Moreover, as computers have become ubiquitous, attacking the computing infrastructure of an organization has emerged as a new form of aggression [2]. Commodity ICs are inexpensive; however, they do not provide any security guarantees whatsoever. Such ICs can potentially have malicious circuitry, popularly known as “Hardware Trojans”. These can effectively bring down the computing infrastructure from the inside.

To further compound the problem, modern systems-on-chips (SOCs) have components designed by different design houses. Although the advantages of this paradigm are many, the provision of accountability is an emerging problem. The chip may fail to perform as expected on field. In such a scenario, perfect accountability – knowing which of the on-chip components are responsible – is desired. Such an accountability solution must be fair to all parties concerned, as being held responsible for a design/security flaw is detrimental to the design house’s credibility in the market.

In this thesis, we apply the principle of redundancy to solve all of the above three problems, which are from three seemingly different domains – reliability, security and accountability. Redundancy is an attractive option because of its generality. The schemes developed are not application or hardware specific. Also, transistors have become cheaper. From costing a few dollars in 1955, the cost of a single transistor has dropped to around a billionth of a dollar in 2014 [3]. The investment made in redundancy is therefore not a big drain on the finances of an organization. The achieved gains far outweigh the costs.

संक्षेप

हालांकि प्रोसेसर डिज़ाइन में पारम्परिक लक्ष्यों जैसे कि, प्रदर्शन, ऊर्जा और क्षेत्रफल, को अभी भी सक्रिय रूप से शोध किया जा रहा है, विश्वसनीयता और सुरक्षा जैसे नए नए क्षेत्रों में भी प्रमुखता आई है।

जैसे जैसे ट्रांजिस्टर छोटे हो गए हैं, नरम त्रुटियों का खतरा बढ़ गया है। विशेष कार्य वाले सिस्टम जो कि कम से कम २-४ पीढ़ी पुरानी चिप्स को उपयोग करने के लिए प्रतिबंधित हैं, और इस कारण कम प्रदर्शन से पीड़ित हैं [१]।

इसके अलावा, कंप्यूटर सर्वव्यापी बन गए हैं, एक संगठन के कंप्यूटिंग अवसंरचना पर हमला करने का एक नया रूप आक्रमकता के रूप में उभरा है [२]। कमोडिटी आईसीएस सस्ती हैं, हालांकि, वे सुरक्षा गारंटी प्रदान नहीं करती हैं। इस तरह के आईसी संभावित रूप से दुर्भावनापूर्ण सर्किटरी हो सकती हैं, जिसे "हार्डवेयर ट्रोजन्स" के रूप में जाना जाता है। ये कंप्यूटिंग ढांचे पर बुरा प्रभाव डालते हैं।

समस्या को और अधिक जटिल बनाने हेतु, आधुनिक सिस्टम-ऑन-चिप्स (एसओसीएस) को विभिन्न रचना घरों द्वारा रचित किया जाता है। हालांकि इस प्रतिमान के कई लाभ भी हैं, परन्तु जवाबदेही का प्रावधान एक उभरती हुई समस्या है। फ़ील्ड पर अपेक्षा के अनुसार चिप निष्पादित करने में विफल हो सकता है। सही जवाबदेही वांछित है। हम यह जानना चाहते हैं कि कौनसा ऑन-चिप घटक जिम्मेदार है। इस तरह का एक जवाबदेही समाधान सभी दलों के लिए उचित होना चाहिए।

इस शोध-प्रबंध में हम उपरोक्त तीन समस्याओं को हल करने के लिए अतिरेक के सिद्धांत को लागू करते हैं, जो तीन अलग-अलग ज्ञानक्षेत्र से प्रतीत होते हैं - विश्वसनीयता, सुरक्षा और जवाबदेही। अपनी व्यापकता के कारण अतिरेकता एक आकर्षक विकल्प है। विकसित योजनाएं एप्लिकेशन या हार्डवेयर विशिष्ट नहीं हैं। साथ ही ट्रांजिस्टर सस्ते हो गए हैं। १९५५ में कुछ डॉलर की लागत से, एक ट्रांजिस्टर की लागत २०१४ में डॉलर के एक अरब हिस्से के करीब आ गई है [३]। अतिरेक में किया गया निवेश एक संगठन के वित्त पर भारी प्रभाव नहीं डाल रहा है। प्राप्त लाभ लागत से अधिक है।

Contents

Certificate	i
Acknowledgements	iii
Abstract	v
1 Introduction	1
1.1 Contributions	4
2 Infrastructure Development: <i>Tejas</i> – A Java based Versatile Micro-architectural Simulator	7
2.1 Introduction	7
2.1.1 Related Work and Motivation	9
2.1.2 Contribution Details	11
2.2 Architecture of <i>Tejas</i>	12
2.2.1 Emulator	14
2.2.2 Transfer Engine	14
2.2.3 Translation Module	17
2.3 Micro-architectural Simulation	22
2.3.1 Semi Event Driven Model	22

2.3.2	Pipelines	22
2.3.3	Branch Predictor	23
2.3.4	Simultaneous Multi-Threading	23
2.3.5	Cache Hierarchy	24
2.3.6	Coherence Protocol	25
2.3.7	Non-Uniform Cache Architectures (NUCA) and DRAM	26
2.3.8	Network-on-Chip (NoC)	27
2.3.9	Power and Temperature Modeling	27
2.3.10	Relaxed Memory Consistency Models	27
2.3.11	Memory Optimizations	28
2.4	Evaluation	30
2.4.1	Validation against Native Hardware	30
2.4.2	Performance	33
2.5	Conclusion	36
3	A Survey of Redundancy Techniques to Provide Reliability in Processors	37
3.1	Introduction	37
3.1.1	Scope	38
3.1.2	Organization	38
3.2	Background	39
3.2.1	Faults in Hardware	39
3.2.2	Taxonomy	41
3.2.3	Checkpoints and Recovery	44
3.3	Circuit and Pipeline Level Techniques	46

3.3.1	Summary	46
3.3.2	Transient Faults	47
3.3.3	Timing Faults	49
3.3.4	Hard Faults	50
3.3.5	Design Faults	50
3.4	Core Based Approaches	51
3.4.1	Summary	51
3.4.2	<i>MultiMaster</i> Approaches	52
3.4.3	<i>SingleSlave</i> – DIVA Family	54
3.4.4	<i>SingleSlave</i> – SlipStream Family	58
3.4.5	<i>MultiSlave</i> Approaches	60
3.5	Thread Level Approaches	62
3.5.1	Summary	62
3.5.2	<i>MultiMaster</i> Schemes – Complete Coverage	64
3.5.3	<i>SingleSlave</i> – <i>Subset</i> Coverage	66
3.6	Multithreaded Programs	67
3.6.1	Summary	67
3.6.2	Uniprocessor Semantics	68
3.6.3	Multiprocessor Semantics - Cache Coherence	71
3.6.4	Multiprocessor Semantics – Memory Consistency	72
3.7	Software Approaches	73
3.7.1	Summary	73
3.7.2	Computation and Dataflow Errors	74

3.7.3	Control Flow Errors	75
3.7.4	All Types of Errors	77
3.7.5	Detecting Faults in Multiprocessors	79
3.8	Conclusion and Future Directions	80
4	Employing Redundancy to Efficiently Detect Soft Errors	81
4.1	Introduction	81
4.2	Related Work and Background	83
4.2.1	Assisted Checking with Full Coverage	84
4.2.2	Independent Checking and Partial Coverage	85
4.2.3	Novelty and Contributions	85
4.3	Motivation	87
4.3.1	Study of Benchmarks: <i>calculix</i> and <i>gcc</i>	87
4.3.2	Overview of the Proposed Scheme	90
4.4	Architecture	93
4.4.1	Description of Physical Structures	94
4.4.2	Overview of Redundant Execution	94
4.4.3	Checking Memory Instructions	96
4.4.4	Rollback Mechanism	98
4.4.5	Resolving Livelock Issues	98
4.4.6	Effect of <i>FluidCheck</i> on the Operating System	99
4.5	Arbiter Logic	101
4.5.1	Mapping a Single Thread	101
4.5.2	Mapping a Set of Threads	101

4.5.3	Thread Migration and NoC Issues	102
4.5.4	Fetch Policies	103
4.6	Evaluation	104
4.6.1	Performance Evaluation	106
4.6.2	Deeper Insights	110
4.6.3	Energy Consumption	115
4.6.4	Sensitivity Studies	117
4.7	Conclusion and Future Work	124
5	Employing Redundancy to Efficiently Detect Hardware Trojan Horses	125
5.1	Introduction	125
5.2	Related Work	128
5.2.1	HTH Prevention Mechanisms	128
5.2.2	HTH Detection Techniques	128
5.3	SecCheck Architecture	131
5.3.1	Hardware Architecture and Attack Model	131
5.3.2	Application Model	132
5.3.3	Verification Strategies	133
5.3.4	Offline Scheduling	135
5.3.5	Execution of an Application	136
5.4	ILP Formulation of the Scheduling Problem	138
5.5	SecCheck Scheduling Algorithm (SSA)	141
5.6	Evaluation	145
5.6.1	<i>SecCheck</i> Overhead	147

5.6.2	SSA Penalty	147
5.6.3	Scalability Analysis	147
5.7	Conclusion	149
6	Employing Redundancy to Efficiently Provide Accountability	151
6.1	Introduction	151
6.2	Related Work	154
6.2.1	Runtime Solutions for Creating Trustworthy Systems	154
6.3	Accountability through Logging	155
6.3.1	Model of a Heterogeneous SoC	155
6.3.2	High Level Logging and Auditing	157
6.3.3	Low Level Logging and Auditing	158
6.4	Auditing Messages in a Sender-Receiver Paradigm	159
6.4.1	The Problem	159
6.4.2	Solution 1): The Parties Themselves do the Auditing	160
6.4.3	Solution 2: Trusted Third Party(TTP) providing a Mail Service	161
6.4.4	Solution 3: Using Embedded Meters	163
6.4.5	Solution 4: Peer Based Auditing	165
6.5	Accountability in 3PIP-containing SoCs	167
6.5.1	Preliminaries	167
6.5.2	Auditing in SoCs with Trusted Hosts	168
6.5.3	Auditing in SoCs with the NoC Designed by an Untrusted Host	169
6.5.4	Auditing in SoCs with a Guest-NoC Trusted by the Host and All Other Guests	177

6.5.5	Auditing in SoCs with a Guest-NoC Untrusted by the Host and All Other Guests	177
6.6	Evaluating the Overheads of Providing Accountability	179
6.6.1	Message sending protocol	180
6.6.2	Design of the Meter	182
6.6.3	Overheads of the Auditing Process	189
6.6.4	Design and Implementation of Redundant Metering: Variant II	191
6.7	Demonstration with Real Bug Scenarios	202
6.7.1	Bug I	203
6.7.2	Bug II	205
6.7.3	Bug III	207
6.8	Other Uses of Reliable Runtime Information	208
6.9	Conclusion	210
7	Conclusion and Future Work	211
	Bibliography	215
	List of Publications	235
	Biography	237

List of Figures

1.1	Employing redundancy based techniques to provide reliability, security and accountability in modern processors (images: aviral.lab.asu.edu, lloydi.com, people-think.biz)	4
2.1	High level architecture of <i>Tejas</i>	13
2.2	Pipelines in <i>Tejas</i>	16
2.3	Translation in <i>Tejas</i>	19
2.4	High level working of a cache (simplified)	24
2.5	Validation : <i>SPEC CPU2006</i> suite	32
2.6	Validation : <i>SPLASH-2</i> suite	32
2.7	Speed – serial workloads	34
2.8	Speed – parallel workloads	35
3.1	<i>Taxonomy of checkers</i>	42
3.2	<i>RAZOR</i>	49
3.3	<i>Dual use of superscalar datapath</i>	49
3.4	NonStop architecture	53
3.5	<i>Basic Idea of DIVA</i>	55
3.6	<i>FastShared</i> Model	56

3.7	<i>MiniDiva</i>	57
3.8	<i>SplitDiva</i>	57
3.9	<i>a) Traditional DIVA b) Hierarchical Verification</i>	57
3.10	<i>Slipstream processors</i>	58
3.11	<i>Coarse-grain verification parallelism</i>	62
3.12	<i>Sphere of Replication</i>	65
4.1	Related work	83
4.2	<i>calculix</i> IPC study	88
4.3	<i>calculix</i> FU hazard study	88
4.4	<i>gcc</i> IPC study	89
4.5	Illustration of the proposed system	91
4.6	Proposed architecture	94
4.7	Leader-checker communication	95
4.8	Mean slowdown - <i>low load</i>	107
4.9	Mean slowdown - <i>medium load</i>	107
4.10	Mean slowdown - <i>high load</i>	108
4.11	Per-benchmark slowdown	110
4.12	Thread mapping snapshot (<i>minIPC</i> , SP-UALF)	110
4.13	Thread mapping snapshot (<i>CRT</i> , SP-UALF)	111
4.14	Per-benchmark slowdown	113
4.15	Energy consumption	116
4.16	Power consumption	116
4.17	Effect of spare cores	118

4.18	Benefit due to cache forwarding	120
4.19	Varying hop latency	121
4.20	Varying NoC topology	122
5.1	<i>SecCheck</i> Hardware Architecture	131
5.2	Application specification and execution in <i>SecCheck</i>	132
5.3	Comparison of different verification techniques	135
6.1	Model of a 3PIP-containing SoC	156
6.2	Candidate solution: sender as auditor	160
6.3	Candidate solution: parties audit themselves	161
6.4	Candidate solutions: using a trusted third party (“M” in the figures)	162
6.5	Candidate solution: Peer auditing schemes	162
6.6	Examples of Different SoC Classes	170
6.7	Effect of misbehavior in the different schemes	171
6.8	Auditing solutions for SoCs with untrusted guest-NoC	178
6.9	Classification of 3PIP-containing SoCs and auditing solutions for each class . . .	179
6.10	Meter design	183
6.11	Hardware architecture	191
6.12	Structure of a meter	193
6.13	Bug I Scenario	203
6.14	Bug II Scenario	205
6.15	Bug III Scenario	207

List of Tables

2.1	<i>Tejas</i> and the Design Space of Architectural Simulators	10
2.2	Comparison of IPC techniques	15
2.3	VISA Instruction Set	18
2.4	Operand Types	18
2.5	Example scenarios requiring <i>corrective operations</i>	26
2.6	Details of the reference hardware	30
2.7	Simulation parameters	31
3.1	Rules for classifying faults	41
3.2	Summary of circuit/pipeline level approaches	47
3.3	Summary of core level approaches	52
3.4	Summary of thread level approaches	63
3.5	Summary of approaches	68
3.6	Summary of approaches	74
4.1	Simulation parameters	105
4.2	Stall statistics and cache hit rates	112
4.3	Stall statistics – across different scheduling policies	113

4.4	Occupancy of Auxiliary Structures	118
4.5	Communication Statistics	119
5.1	Parameters in the ILP formulation	138
5.2	Variables in the ILP formulation	139
5.3	Additional variables in the ILP formulation	139
5.4	Parameters of the synthetic task graphs	145
5.5	<i>SecCheck</i> overhead ($((\frac{PM_{secureILP}}{PM_{insecureILP}} - 1) \times 100)$)	146
5.6	SSA penalty ($((\frac{PM_{SSA}}{PM_{secureILP}} - 1) \times 100)$)	146
5.7	SSA average run time (seconds) for different task graph sizes	148
6.1	Payoff rules for the different schemes	172
6.2	Payoff matrix for the game “Redundant metering”	172
6.3	Payoff matrix for the game “single meter at the host”	175
6.4	Payoff matrix for the game “Single meter at the guest”	176
6.5	Message sending protocol	181
6.6	Meter area estimation	184
6.7	Simulation parameters	189
6.8	Accelerator types and simulated workloads	190
6.9	Description of the logs	192
6.10	Guest resource request protocol	197
6.11	Simulation parameters	199
6.12	ASIC v/s software implementations of accelerators	200
6.13	Area estimation of auxiliary structures	202

6.14 Logging different packets related to the USB protocol	206
6.15 Need for Runtime Information Regarding Different On-Chip Components . . .	209