

# Embedded Processor-Based Self-Test

Dimitris Gizopoulos  
Antonios Paschalis  
Yervant Zorian



Kluwer Academic Publishers

TP31  
G539

# EMBEDDED PROCESSOR-BASED SELF-TEST

by

DIMITRIS GIZOPOULOS

*University of Piraeus, Piraeus, Greece*

ANTONIS PASCHALIS

*University of Athens, Athens, Greece*

and

YERVANT ZORIAN

*Virage Logic, Fremont, California, U.S.A.*



E200601016

**KLUWER ACADEMIC PUBLISHERS**

BOSTON / DORDRECHT / LONDON

A C.I.P. Catalogue record for this book is available from the Library of Congress.

ISBN 1-4020-2785-0 (HB)

ISBN 1-4020-2801-6 (e-book)

---

Published by Kluwer Academic Publishers,  
P.O. Box 17, 3300 AA Dordrecht, The Netherlands.

Sold and distributed in North, Central and South America  
by Kluwer Academic Publishers,  
101 Philip Drive, Norwell, MA 02061, U.S.A.

In all other countries, sold and distributed  
by Kluwer Academic Publishers,  
P.O. Box 322, 3300 AH Dordrecht, The Netherlands.

*Printed on acid-free paper*



All Rights Reserved

© 2004 Kluwer Academic Publishers, Boston

No part of this work may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, microfilming, recording or otherwise, without written permission from the Publisher, with the exception of any material supplied specifically for the purpose of being entered and executed on a computer system, for exclusive use by the purchaser of the work.

Printed in the Netherlands.

## EMBEDDED PROCESSOR-BASED SELF-TEST

# FRONTIERS IN ELECTRONIC TESTING

*Consulting Editor*  
**Vishwani D. Agrawal**

## ***Books in the series:***

### **Embedded Processor-Based Self-Test**

D. Gizopoulos  
ISBN: 1-4020-2785-0

### **Testing Static Random Access Memories**

S. Hamdioui  
ISBN: 1-4020-7752-1

### **Verification by Error Modeling**

K. Radecka and Zilic  
ISBN: 1-4020-7652-5

### **Elements of STIL: Principles and Applications of IEEE Std. 1450**

G. Maston, T. Taylor, J. Villar  
ISBN: 1-4020-7637-1

### **Fault Injection Techniques and Tools for Embedded systems Reliability Evaluation**

A. Benso, P. Prinetto  
ISBN: 1-4020-7589-8

### **High Performance Memory Memory Testing**

R. Dean Adams  
ISBN: 1-4020-7255-4

### **SOC (System-on-a-Chip) Testing for Plug and Play Test Automation**

K. Chakrabarty  
ISBN: 1-4020-7205-8

### **Test Resource Partitioning for System-on-a-Chip**

K. Chakrabarty, Iyengar & Chandra  
ISBN: 1-4020-7119-1

### **A Designers' Guide to Built-in Self-Test**

C. Stroud  
ISBN: 1-4020-7050-0

### **Boundary-Scan Interconnect Diagnosis**

J. de Sousa, P. Cheung  
ISBN: 0-7923-7314-6

### **Essentials of Electronic Testing for Digital, Memory, and Mixed Signal VLSI Circuits**

M.L. Bushnell, V.D. Agrawal  
ISBN: 0-7923-7991-8

### **Analog and Mixed-Signal Boundary-Scan: A Guide to the IEEE 1149.4 Test Standard**

A. Osseiran  
ISBN: 0-7923-8686-8

### **Design for At-Speed Test, Diagnosis and Measurement**

B. Nadeau-Dosti  
ISBN: 0-79-8669-8

### **Delay Fault Testing for VLSI Circuits**

A. Krstic, K.-T. Cheng  
ISBN: 0-7923-8295-1

### **Research Perspectives and Case Studies in System Test and Diagnosis**

J.W. Sheppard, W.R. Simpson  
ISBN: 0-7923-8263-3

### **Formal Equivalence Checking and Design Debugging**

S.-Y. Huang, K.-T. Cheng  
ISBN: 0-7923-8184-X

### **Defect Oriented Testing for CMOS Analog and Digital Circuits**

M. Sachdev  
ISBN: 0-7923-8083-5

to Georgia, Dora and Rita

## Preface

This book discusses self-testing techniques in embedded processors. These techniques are based on the execution of test programs aiming to lower the cost of testing for processors and surrounding blocks.

Manufacturing test cost is already a dominant factor in the overall development cost of Integrated Circuits (IC). Consequently, cost effective methodologies are continuously sought for test cost reduction. Self-test, the ability of a circuit to test itself is a widely adopted Design for Test (DfT) methodology. It does not only contribute to the test cost reduction but also improves the quality of test because it allows a test to be performed at the actual speed of the device, to detect defect mechanisms that manifest themselves as delay malfunctions. Furthermore, self-test is a re-usable test solution. It can be activated several times throughout the device's life-cycle. The self-testing infrastructure of a chip can be used to detect latent defects that do not exist at manufacturing phases, but they appear during the chip's operating life

The application of self-testing, as well as, other testing methods, face serious challenges when the circuit under test is a processor. This is due to the fact that processor architectures are particularly sensitive to performance degradation due to extensive design changes for testability improvement. DfT modifications of a circuit, including those that implement self-testing, usually lead to area, performance and power consumption overheads that may not be affordable in a processor design. Processor testing and self-testing is a particularly challenging problem due to sophisticated complex processor structure, but it is also a very important problem that needs special attention because of the central role that processors play in every electronic system.

In this book, an emerging self-test methodology that recently captured the interest of test technologists is studied. *Software-based self-testing*, also called *processor-based self-testing*, takes advantage of the programmability of processors and allows them to test themselves with the effective execution of embedded self-test programs. Moreover, software-based self-testing takes advantage of the accessibility that processors have to all other surrounding

blocks of complex designs to test these blocks as well with such self-test programs. The already established System-on-Chip design paradigm that is based on pre-designed and pre-verified embedded cores employs one or more embedded processors of different architectures. Software-based self-testing is a very suitable methodology for manufacturing and in-field testing of embedded processors and surrounding blocks.

In this book, software-based self-testing is described, as a practical, low-cost, easy-to-apply self-testing solution for processors and SoC designs. It relaxes the tight relation of manufacturing testing with high-performance, expensive IC test equipment and hence results in test cost reduction. If appropriately applied, software-based self-testing can reach a very high test quality (high fault coverage) with reasonable test engineering effort, small test development cost and short test application time.

Also, this book sets a basis for comparisons among different software-based self-testing techniques. This is achieved by: describing the basic requirements of this test methodology; focusing on the basic parameters that have to be optimized; and applying it to a set of publicly available benchmark processors with different architectures and instruction sets.

Dimitris Gizopoulos	<i>Piraeus, Greece</i>
Antonis Paschalis	<i>Athens, Greece</i>
Yervant Zorian	<i>Fremont, CA, USA</i>



## Acknowledgments

The authors would like to acknowledge the support and encouragement by Dr. Vishwani D. Agrawal, the *Frontiers in Electronic Testing* book series consulting editor. Special thanks are also due to Carl Harris and Mark de Jongh of Kluwer Academic Publishers for the excellent collaboration in the production of this book.

The authors would like to acknowledge the help and support of several individuals at the University of Piraeus, the University of Athens and Virage Logic and in particular the help of Nektarios Kranitis and George Xenoulis.

# CONTENTS

Contents	v
List of Figures	viii
List of Tables	ix
Preface	xiii
Acknowledgments	xv
1. INTRODUCTION	1
1.1 Book Motivation and Objectives	1
1.2 Book Organization	4
2. DESIGN OF PROCESSOR-BASED SOC	7
2.1 Integrated Circuits Technology	7
2.2 Embedded Core-Based System-on-Chip Design	8
2.3 Embedded Processors in SoC Architectures	11
3. TESTING OF PROCESSOR-BASED SOC	21
3.1 Testing and Design for Testability	21
3.2 Hardware-Based Self-Testing	32
3.3 Software-Based Self-Testing	41
3.4 Software-Based Self-Test and Test Resource Partitioning	46
3.5 Why is Embedded Processor Testing Important?	48
3.6 Why is Embedded Processor Testing Challenging?	49
4. PROCESSOR TESTING TECHNIQUES	55
4.1 Processor Testing Techniques Objectives	55
4.1.1 External Testing versus Self-Testing	56
4.1.2 DFT-based Testing versus Non-Intrusive Testing	57
4.1.3 Functional Testing versus Structural Testing	58
4.1.4 Combinational Faults versus Sequential Faults Testing	59
4.1.5 Pseudorandom versus Deterministic Testing	60
4.1.6 Testing versus Diagnosis	62
4.1.7 Manufacturing Testing versus On-line/Field Testing	63
4.1.8 Microprocessor versus DSP Testing	63
4.2 Processor Testing Literature	64
4.2.1 Chronological List of Processor Testing Research	64
4.2.2 Industrial Microprocessors Testing	78
4.3 Classification of the Processor Testing Methodologies	78
5. SOFTWARE-BASED PROCESSOR SELF-TESTING	81
5.1 Software-based self-testing concept and flow	82
5.2 Software-based self-testing requirements	87
5.2.1 Fault coverage and test quality	88
5.2.2 Test engineering effort for self-test generation	90

5.2.3	Test application time	91
5.2.4	A new self-testing efficiency measure	96
5.2.5	Embedded memory size for self-test execution	97
5.2.6	Knowledge of processor architecture	98
5.2.7	Component based self-test code development	99
5.3	Software-based self-test methodology overview	100
5.4	Processor components classification	107
5.4.1	Functional components	108
5.4.2	Control components	111
5.4.3	Hidden components	112
5.5	Processor components test prioritization	113
5.5.1	Component size and contribution to fault coverage	115
5.5.2	Component accessibility and ease of test	117
5.5.3	Components' testability correlation	119
5.6	Component operations identification and selection	121
5.7	Operand selection	124
5.7.1	Self-test routine development: ATPG	125
5.7.2	Self-test routine development: pseudorandom	133
5.7.3	Self-test routine development: pre-computed tests	137
5.7.4	Self-test routine development: style selection	139
5.8	Test development for processor components	141
5.8.1	Test development for functional components	141
5.8.2	Test development for control components	141
5.8.3	Test development for hidden components	143
5.9	Test responses compaction in software-based self-testing	146
5.10	Optimization of self-test routines	148
5.10.1	"Chained" component testing	149
5.10.2	"Parallel" component testing	152
5.11	Software-based self-testing automation	153
6	CASE STUDIES – EXPERIMENTAL RESULTS	157
6.1	Parwan processor core	158
6.1.1	Software-based self-testing of Parwan	159
6.2	Plasma/MIPS processor core	160
6.2.1	Software-based self-testing of Plasma/MIPS	163
6.3	Meister/MIPS reconfigurable processor core	168
6.3.1	Software-based self-testing of Meister/MIPS	170
6.4	Jam processor core	171
6.4.1	Software-based self-testing of Jam	172
6.5	oc8051 microcontroller core	173
6.5.1	Software-based self-testing of oc8051	175
6.6	RISC-MCU microcontroller core	176
6.6.1	Software-based self-testing of RISC-MCU	177

6.7	oc54x DSP Core	178
6.7.1	Software-based self-testing of oc54x	179
6.8	Compaction of test responses	181
6.9	Summary of Benchmarks	181
7.	PROCESSOR-BASED TESTING OF SOC	185
7.1	The concept	185
7.1.1	Methodology advantages and objectives	188
7.2	Literature review	190
7.3	Research focus in processor-based SoC testing	193
8.	CONCLUSIONS	195
	References	197
	Index	213
	About the Authors	217

## LIST OF FIGURES

Figure 2-1: Typical System-on-Chip (SoC) architecture. _____	9
Figure 2-2: Core types of a System-on-Chip. _____	11
Figure 3-1: ATE-based testing. _____	28
Figure 3-2: Self-testing of an IC. _____	34
Figure 3-3: Self-testing with a dedicated memory. _____	38
Figure 3-4: Self-testing with dedicated hardware. _____	39
Figure 3-5: Software-based self-testing concept for processor testing. _____	42
Figure 3-6: Software-based self-testing concept for testing a SoC core. _____	43
Figure 5-1: Software-based self-testing for a processor (manufacturing). _____	82
Figure 5-2: Software-based self-testing for a processor (periodic). _____	84
Figure 5-3: Application of software-based self-testing: the three steps. _____	86
Figure 5-4: Engineering effort (or cost) versus fault coverage. _____	91
Figure 5-5: Test application time as a function of the K/W ratio. _____	94
Figure 5-6: Test application time as a function of the $f_{up}/f_{tester}$ ratio. _____	95
Figure 5-7: Software-based self-testing: overview of the four phases. _____	102
Figure 5-8: Phase A of software-based self-testing. _____	103
Figure 5-9: Phase B of software-based self-testing. _____	104
Figure 5-10: Phase C of software-based self-testing. _____	105
Figure 5-11: Phase D of software-based self-testing. _____	107
Figure 5-12: Classes of processor components. _____	108
Figure 5-13: Prioritized component-level self-test program generation. _____	114
Figure 5-14: ALU component of the MIPS-like processor. _____	122
Figure 5-15: ATPG test patterns application from memory. _____	129
Figure 5-16: ATPG test patterns application with immediate instructions. _____	131
Figure 5-17: Forwarding logic multiplexers testing. _____	145
Figure 5-18: Two-step response compaction. _____	147
Figure 5-19: One-step response compaction. _____	147
Figure 5-20: "Chained" testing of processor components. _____	150
Figure 5-21: "Parallel" testing of processor components. _____	153
Figure 5-22: Software-based self-testing automation. _____	154
Figure 7-1: Software-based self-testing for SoC. _____	186

## LIST OF TABLES

Table 2-1: Soft, firm and hard IP cores.	10
Table 2-2: Embedded processor cores (1 of 3).	15
Table 2-3: Embedded processor cores (2 of 3).	16
Table 2-4: Embedded processor cores (3 of 3).	17
Table 4-1: External testing vs. self-testing.	57
Table 4-2: DfT-based vs. non-intrusive testing.	57
Table 4-3: Functional vs. structural testing.	59
Table 4-4: Combinational vs. sequential testing.	60
Table 4-5: Pseudorandom vs. deterministic testing.	62
Table 4-6: Testing vs. diagnosis.	63
Table 4-7: Manufacturing vs. on-line/field testing.	63
Table 4-8: Processor testing methodologies classification.	79
Table 5-1: Operations of the MIPS ALU.	124
Table 5-2: ATPG-based self-test routines test application times (case 1).	132
Table 5-3: ATPG-based self-test routines test application times (case 2).	132
Table 5-4: Characteristics of component self-test routines development.	139
Table 6-1: Parwan processor components.	159
Table 6-2: Self-test program statistics for Parwan.	160
Table 6-3: Fault simulation results for Parwan processor.	160
Table 6-4: Plasma processor components.	161
Table 6-5: Plasma processor synthesis for Design I.	162
Table 6-6: Plasma processor synthesis for Design II.	162
Table 6-7: Plasma processor synthesis for Design III.	163
Table 6-8: Fault simulation results for the Plasma processor Design I.	164
Table 6-9: Self-test routine statistics for Designs II and III of Plasma.	164
Table 6-10: Fault simulation results for Designs II and III of Plasma.	165
Table 6-11: Plasma processor synthesis for Design IV.	167
Table 6-12: Comparisons between Designs II and IV of Plasma.	167
Table 6-13: Meister/MIPS processor components.	168
Table 6-14: Meister/MIPS processor synthesis.	169
Table 6-15: Self-test routines statistics for Meister/MIPS processor.	170
Table 6-16: Fault simulation results for Meister/MIPS processor.	170
Table 6-17: Jam processor components.	171
Table 6-18: Jam processor synthesis.	172
Table 6-19: Self-test routine statistics for Jam processor.	173
Table 6-20: Fault simulation results for Jam processor.	173
Table 6-21: oc8051 processor components.	174
Table 6-22: oc8051 processor synthesis.	174

Table 6-23: Self-test routine statistics for oc8051 processor.	175
Table 6-24: Fault simulation results for oc8051 processor.	176
Table 6-25: RISC-MCU processor components.	176
Table 6-26: RISC-MCU processor synthesis.	177
Table 6-27: Self-test routine statistics for RISC-MCU processor.	178
Table 6-28: Fault simulation results for RISC-MCU processor.	178
Table 6-29: oc54x processor components.	179
Table 6-30: oc54x DSP synthesis.	179
Table 6-31: Self-test routines statistics for oc54x DSP.	180
Table 6-32: Fault simulation results for oc54x DSP.	180
Table 6-33: Execution times of self-test routines.	181
Table 6-34: Summary of benchmark processor cores.	182
Table 6-35: Summary of application of software-based self-testing.	183

# Chapter 1

## *Introduction*

---

### **1.1 Book Motivation and Objectives**

Electronic products are used today in the majority of our daily activities. Thus, they enabled efficiency, productivity, enjoyment and safety.

The Integrated Circuits (ICs) realized today consist of multiple millions of logic gates and even more memory cells. They are implemented in, very deep sub-micron (VDSM) process technologies and often consist of multiple, pre-designed entities called Intellectual Property (IP) cores. This design methodology that allowed the integration of embedded IP cores is known as Embedded Core-Based System-on-Chip (SoC) design methodology. SoC design flow supported by appropriate Computer Aided Design (CAD) tools has dramatically improved design productivity and has opened up new horizons for successful implementation of sophisticated chips.

An important role in the architecture of complex SoC is played by embedded processors. Embedded processors and other cores built around them constitute the basic functional elements of today's SoCs in embedded systems. Embedded processors have optimized design (in terms of silicon area, performance, power consumption, etc), and provide the means for the integration of sophisticated, flexible, upgradeable and re-configurable functionality of a complex SoC. In many cases, more than one embedded



processors exist in a SoC, each of which takes over different tasks of the system and sharing the processing workload.

Issues such as the quality of the final SoC, the reliability of the manufactured ICs, and the reduced possibility of delivering malfunctioning chips to the end users, are rapidly getting more importance today with the increasing criticality of most of electronic systems applications.

In the context of these quality and reliability requirements, complex SoC designs, realized in dense manufacturing technologies face serious problems that need special consideration. Manufacturing test of complex chips based on external Automatic Test Equipment (ATE), as a method to guarantee that the delivered chips are correctly operating, is becoming less feasible and more expensive than ever. The volume of test data that must be applied to each manufactured chip is becoming very large, the test application time is increasing and the overall manufacturing test cost is becoming the dominant part of the total chip development cost.

Under these circumstances, which are expected to get worse as circuits size shrinks and density increases, the effective migration of the manufacturing test resources from outside of the chip (ATE) to on-chip, built-in resources and thus the effective replacement of external based testing with internally executed *self-testing* is, today the test technology of choice for all SoCs in practice. Self-testing allows at-speed testing, i.e. test execution at the actual operating speed of the chip. Thus all physical faults that cause either timing miss-behavior or an incorrect binary value can be detected. Also, self-testing drastically reduces test data storage requirements and test application time, both of which explode when external, ATE-based testing is used. Therefore, the extensive use of self-testing has a direct impact on the reduction of the overall chip test cost.

Testing of processors or microprocessors, even when they are not deeply embedded in a complex system, is known to be a challenging task itself. Classical testing approaches used in other digital circuits are not adequate to the carefully optimized processor designs, because they can't reach the same efficiency as in other types of digital circuits. Also, self-test approaches, successfully used to improve the testability of digital circuits, are not very suitable for processor testing because such techniques usually add overheads in the processor's performance, silicon area, pin count and power consumption. These overheads are often not acceptable for processors which have been specifically optimized to satisfy very strict area, speed and power consumption requirements.

This book primarily discusses the special problem of testing and self-testing of embedded processors in SoC architectures, as well as the problem of testing and self-testing other cores of the SoC using the embedded processor as test infrastructure.