SOFTWARE-BASED SELF-TESTING FOR A RISC PROCESSOR

TEH WEE MENG

UNIVERSITI TEKNOLOGI MALAYSIA

SOFTWARE-BASED SELF-TESTING FOR A RISC PROCESSOR

TEH WEE MENG

A project submitted in partial fulfilment of the
requirements for the award of the degree of
Master of Engineering (Electrical - Computer and Microelectronic System)

Faculty of Electrical Engineering
Universiti Teknologi Malaysia

JUNE 2014

Specially dedicated to my family, lecturers and friends who have guided, motivated and inspired me throughout my journey of education

ACKNOWLEDGEMENT

Firstly, I would like to thank my project supervisor Dr. Ooi Chia Yee for all her guidance and advices throughout the duration of this project. This project would not have been a successful one without her.

I also wish to express my gratitude to the authors of all technical papers listed in the references section that I have been referring to, which have provided me with all the necessary knowledge and concepts in completing this project.

Special thanks to all friends and anyone who have helped me in one way or another during this project. Thanks to my family also for all the supports given to me throughout the years of my studies.

ABSTRACT

Software-based self-testing (SBST) has been touted as the effective way to test the processors effectively, with reasonable test coverage, plus the advantages of at-speed testing, and without performance degradation in terms of area and power. Previous work has been done on combining SBST with partial scan logic insertion at Register Transfer Language (RTL) level for a 16-bit RISC processor design. In this project, focus will be done on test coverage improvement without the use of scan logic.

ABSTRAK

Perisian ujian sendiri (SBST) telah disebut-sebut sebagai cara yang berkesan untuk menguji unit pemprosesan pusat (CPU), dengan liputan ujian yang munasabah, pelbagai manfaat seperti ujian-sama-kelajuan, tiada kesan negatif ke atas prestasi dari segi pembaziran keluasan and tenaga. Kerja sebelumnya telah dilakukan ke atas menggabungkan SBST dengan penggunaan separa logik imbasan di dalam RTL untuk sejenis 16-bit RISC CPU. Dalam projek ini, tumpuan diberikan terhadap peningkatan liputan ujian tanpa menggunakan logik imbasan.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

SBST - Software-Based Self-Testing

RISC - Reduced Instruction Set Computer

CPU - Central Processing Unit

CISC - Complex Instruction Set Computer

I/O - Input/Output

ROM - Read Only Memory

ISA - Instruction Set Architecture

IP - Intellectual Properties

ALU - Arithmetic Logic Unit

RTL - Register Transfer Language

ATPG - Automatic Test Pattern Generation

DC - Design Compiler

STIL - Standard Test Interface Language

ATE - Automatic Test Equipment

BIST - Built-In Self-Test

IC - Integrated Circuit

SOC - System On Chip

CU - Control Unit

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

This project is about implementing the technique of Software-Based Self-Testing (SBST) on a Reduced Instruction Set Computer (RISC) processor. Effectiveness of this testing method will be judged on the achieved test coverage, test program size and testing cycle count. Comparison with previous work that combines SBST and partial scan insertion technique would be done. This chapter gives a brief introduction of project background, objective, scope, implementation plan and problem statement, as well as the organization of this thesis.

1.1 Background

In this electronic age of 21^{st} century, processors, also known as Central Processing Unit (CPU), are the heart of almost all smart electronics devices, especially the ubiquitous smartphones and tablets. The word “smart” stemmed from the fact that processors are continuously making all the smart decisions in terms of the controls and instruction executions in a complex system.

RISC stands for Reduced Instruction Set Computer. This is a CPU design strategy hinged on the concept that simplified instructions can provide higher performance, with faster execution of each instruction. The term "reduced" pinpoints the fact that the amount of time required by any single instruction to accomplish its task is reduced - at most a single data memory cycle. On the other hand, "complex instructions" executed by CISC (Complex Instruction Set Computer) CPUs may require dozens of data memory cycles in order to execute a single instruction. A quick comparison between RISC and CISC is shown in Table 1.1. In particular, RISC processors typically have separate instructions for I/O and data processing.

Table 1.1 : Comparison between CISC and RISC processor

Design Feature	CISC Processor	RISC Processor
Instruction length	Variable length	Fixed length
Addressing modes	Many	Few
Clock cycles	Complex instructions, each instruction requires many clock cycles to execute	Simple instructions, typically 1 clock cycle for each instruction
Memory access	Support memory-to-memory instructions	Only load & store instructions have access to memory
Registers	Small numbers of general purpose registers	A large register file, to be used for any purpose (prevents large interactions with memory)
Control Unit	Use large microcoded ROM	Hardwired
Pipelining	Lousy pipelining	Pipelined datapath

RISC processors usage ranges from cellular phones, tablets to supercomputers. Examples include ARM architecture, MIPS line, Hitachi’s SuperH, Atmel AVR, SPARC by Oracle, IBM’s Power Architecture, Hewlett-Packard’s PA-

RISC, Alpha and so on. As shown in Figure 1.1, RISC designs normally adopt Harvard architecture, whereby instruction and data memories are separate modules.

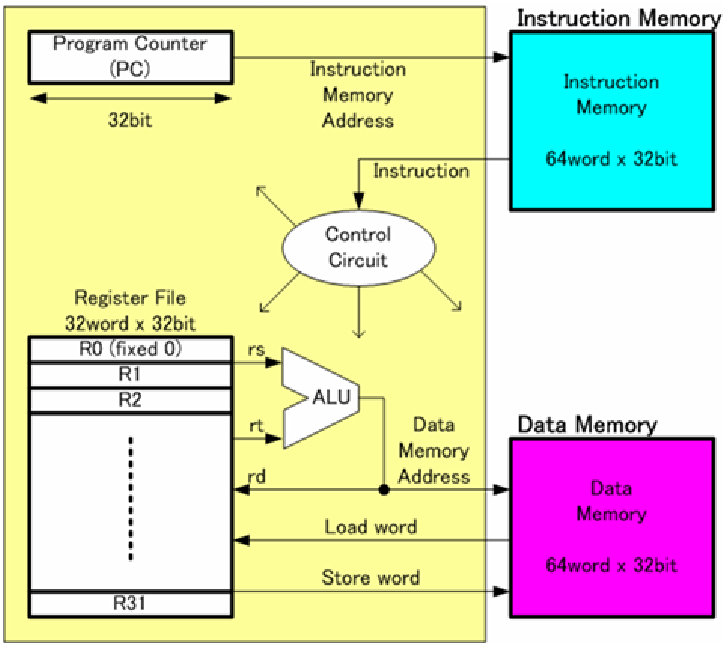

Figure 1.1 : A typical RISC processor design

Basic building blocks of a typical RISC processor include:

- Instruction Set Architecture (ISA) registers.
- Fundamental IPs such as:
  - ALU – implementation of data processing instructions with arithmetic (add, subtract & multiply) and logical operations.
  - Memory access unit – manipulation of both data and instruction memory addresses.
- Control/steering logic/pipeline registers.
  - Control Unit – a module that extracts instructions from memory, decodes and executes them, calling on the ALU when necessary.
  - Pipeline registers – registers that store address of current executed instruction, handles interrupt and backup return address.
- Pipeline-related control logic.

Generally, RISC processor’s instruction can be divided into four categories of:

(a) CPU control – instructions such as NOP, STOP, SET and CLR which do not generate numeric results but alter processor’s state. SET and CLR allow the set and clear operation of any status or control registers.
(b) Data transfer – instructions of MOV, LOAD and PUSH that copy the content of an internal register to another register, a memory location, or load the data from these sources to register file.
(c) Branch and subroutine – instructions like JMP and BRC that alter the value of program counter and access the call stack.
(d) Arithmetic and Logic – instructions of ADD, NEG, XOR that generate numeric results as a function of two source operands.

1.2 Objective

The main objective of this project is to understand the design of a 16-bit RISC processor from previous work, and apply SBST methodology that generates the constrained test patterns to test the processor effectively.

1.3 Scope

The scope in this project involves understanding the Verilog design of a RISC processor (reused from previous student’s project), verifying the functionality of the RISC processor through RTL simulation, synthesizing the Verilog codes into gate level netlist, generating test patterns for the processor with a set of constraints, and performing test coverage analysis.

The three main tools that are used during development of this project are summarized as below:

- Quartus II (version 11.0) is used for Verilog coding analysis/modifications and RTL simulation.

- Design Compiler (DC) is used for synthesis of a RTL into its gate-level netlist.
- Tetramax is the Automatic Test Pattern Generation (ATPG) tool used for test pattern generation and test coverage analysis.

1.4 Problem Statement

The previous work [1] that combines SBST technique and partial scan insertion technique has the limitations of test time generation and area overhead. This project is to eliminate area overhead incurred from the partial scan technique while retaining the reasonable test coverage.

1.5 Research Methodology

As indicated in Figure 1.2, this project shall start with formulating remaining problem of previous SBST related work. Literature study on the RISC processor design and SBST methodology follows, which would require understanding of Verilog codes and getting familiarized with the use of synthesis and ATPG tools. Experiments start as soon as project approach is finalized, and project will end with thesis writing.

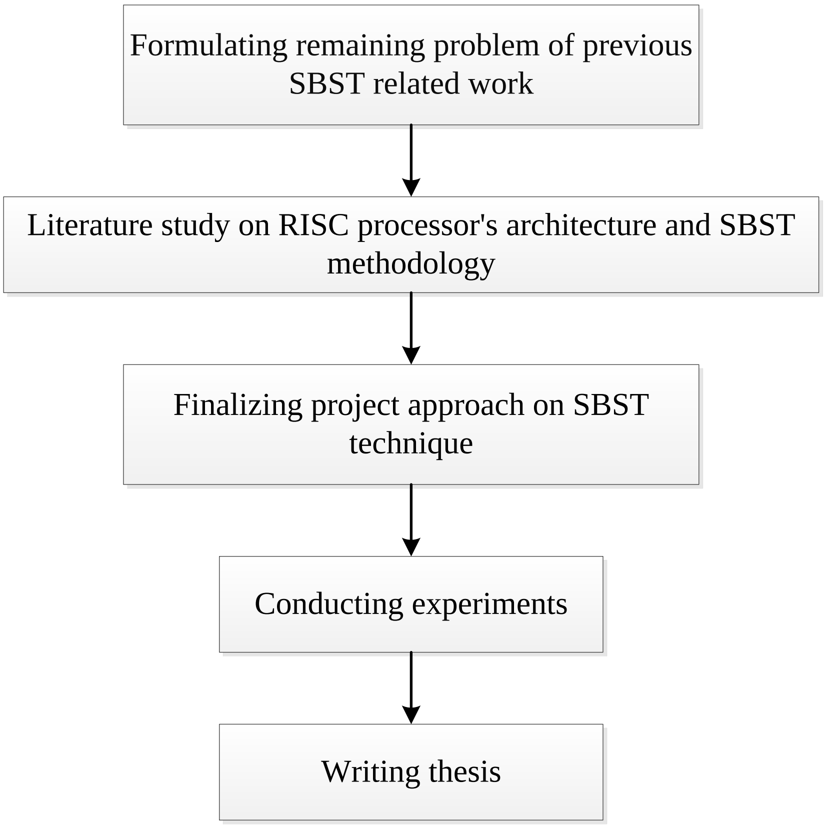

Figure 1.2 : Flow chart of research methodology

1.6 Thesis Outline

This thesis is separated into five chapters. It starts with introduction in Chapter 1, which covers the project background, objective, scope and problem statement. This is followed by Chapter 2, which comprises the literature review of various processors’ testing methods and the architecture of the RISC processor used for the project. Chapter 3 outlines the methodology and implementation in this project. Chapter 4 summarizes all the ATPG results and discussion. Chapter 5 gives the conclusion and recommendations for future work of this project.

REFERENCES

1. Ang Kim Chuan, “Software-Based Self-Test with Scan Design at Register Transfer Level for 16-bit RISC Processor”, Master Thesis of Universiti Teknologi Malaysia, Skudai, 2010.

2. Nektarios Kranitis, Antonis Paschalis, Dimitris Gizopoulos, and George Xenoulis, “Software-Based Self-Testing of Embedded Processors”, IEEE Trans. Comput., vol. 54, no. 4, pp. 461-475, Apr. 2005.

3. L. Chen and S. Dey, “Software-Based Self-Testing Methodology for Processor Cores”, IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 20, no. 3, pp. 369-380, Mar. 2001.

4. A. Krstic, L. Chen, W.C. Lai, K.T. Cheng, and S. Dey, “Embedded Software-Based Self-Test for Programmable Core-Based Designs”, IEEE Design and Test of Computers, pp. 18-26, July/Aug. 2002.

5. Chung-Ho Chen, Chih-Kai Wei, Tai-Hua Lu, and Hsun-Wei Gao, “Software-Based Self-Testing With Multiple-Level Abstractions for Soft Processor Cores”, IEEE Trans. VLSI Syst., vol. 15, no. 5, May 2007.

6. Prof. Bruce Jacob, “The RiSC-16 Instruction Set Architecture”, Fall 2000.

7. Psarakis, M., Gizopoulos, D., Sanchez, E., and Reorda, M.S., “Microprocessor Software-Based Self-Testing”, *IEEE Design and Test of Computers*, pp. 4-19, May-June 2010.

8. Anuruddh Sharma and Mukti Awad, “A 16-bit RISC Processor For Computer Hardware Introduction”, IRACST – Engineering Science and Technology: An International Journal (ESTIJ), ISSN: 2250-3498, vol. 2, no. 3, Jun. 2012.

9. L. Chen, S. Ravi, A. Raghunathan, and S. Dey, “A scalable software-based self-test methodology for programmable processors,” in Proc. 17th Design Autom. Conf., pp. 548–553, 2003.