FUNCTIONAL TEST GENERATION USING MICRO OPERATION FAULT MODEL

ONG HUI YIEN

A project report 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

MAY 2011
To my beloved parents, wife and son
ACKNOWLEDGEMENT

This project would not have been successful without the endless support from many great people. Firstly, I would like to express my highest gratitude to my project supervisor, Dr. Ooi Chia Yee for her continuous guidance, patience and support for me throughout the project.

I would also like to take this opportunity to thank my fellow colleagues in Intel, Jonie Lim and Yoong Yaw for their guidance and knowledge sharing in SystemVerilog language and VCS tool. With their help, I was able to build the correct constraint model, generate quality sets of test pattern and this leads to better analysis in this project. Besides, I would also like to thank my peers, Alice Koh, Kevin Leong, Wee Soon and Si Long for their help and encouragement.

Lastly and most importantly, thanks to my wife and son who have been very supportive, patient and encouraging throughout this project.
ABSTRACT

As semiconductor technology advances further into nanometer regime, integrated circuit testing and validation continues to play a very important role to ensure high quality product. Conventionally, test patterns are generated from a gate level netlist using test generation tool. However, as the digital design increases in complexity, the gate level test generation process becomes more complicated and time consuming. As an extended alternative to this, functional fault model like micro operation fault model was introduced. However, in order to implement this, proper automation is necessary while minimizing intensive manual labor. Unfortunately, currently there is only proprietary version of automation available. In this project, an automated platform to generate test pattern using micro operation fault model was built using Perl programming language. The methodology involves conversion of behavioral model of design under test into extended finite state machine. This is followed by micro operation fault detection, fault activation and fault propagation with all the corresponding constraint sequences captured and converted into constraint model using SystemVerilog, a hardware description language. These models of fault free and intended faulty circuit were fed into a constraint solver tool, VCS by Synopsys to generate the test pattern. For verification purpose, these test patterns were validated by simulating the circuit using Altera Quartus II tool. The result of this project shows that reasonable fault coverage was achieved using this methodology.
ABSTRAK

# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>CHAPTER</th>
<th>TITLE</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>DECLARATION</td>
<td>ii</td>
<td></td>
</tr>
<tr>
<td>DEDICATION</td>
<td>iii</td>
<td></td>
</tr>
<tr>
<td>ACKNOWLEDGEMENT</td>
<td>iv</td>
<td></td>
</tr>
<tr>
<td>ABSTRACT</td>
<td>v</td>
<td></td>
</tr>
<tr>
<td>ABSTRAK</td>
<td>vi</td>
<td></td>
</tr>
<tr>
<td>TABLE OF CONTENTS</td>
<td>vii</td>
<td></td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td>ix</td>
<td></td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>LIST OF ABBREVIATIONS</td>
<td>xii</td>
<td></td>
</tr>
<tr>
<td>LIST OF APPENDICES</td>
<td>xiii</td>
<td></td>
</tr>
</tbody>
</table>

## 1 INTRODUCTION

1.1 Project Background
1
1.2 Problem Statement
3
1.3 Objectives
3
1.4 Scopes of Study
4
1.5 Organization of the Report
5

## 2 LITERATURE REVIEW

2.1 High Level Test Generation Techniques
6
2.2 Functional Fault Model
8
2.3 Extended Finite State Machine
9
2.4 Fummi’s Fault Model
10
2.5 Chen’s Fault Model
10
3 METHODOLOGY

3.1 Project Planning 13
3.2 Methodology for Project Design and Implementation 16
  3.2.1 EFSM Modeling of DUT 17
  3.2.2 Perl Programming 18
  3.2.3 Constraint Sequence Generation 20
  3.2.4 Constraint Model Generation 21
  3.2.5 SystemVerilog Test Bench 22
  3.2.6 Test Pattern Generation and Validation 24

4 DEVELOPMENT OF FUNCTIONAL TEST GENERATION PLATFORM AND TEST BENCH DESIGN

4.1 Platform Development Overview 27
4.2 Software Design Architecture 28
  4.2.1 VHDL Processing and Equivalent EFSM Modeling 28
  4.2.2 Fault Activation, Fault Propagation, and Test Sequence Generation 35
  4.2.3 Constraint Models Generation 37
4.3 SystemVerilog Test Bench Design 39

5 RESULT AND DISCUSSION

5.1 Result and Validation 45
5.2 Result Analysis 48

6 CONCLUSION AND RECOMMENDATION

6.1 Challenges Encountered and Conclusion 50
6.2 Recommendation for Future Work 52

REFERENCES 54

APPENDIX A 56
## LIST OF TABLES

<table>
<thead>
<tr>
<th>TABLE NO.</th>
<th>TITLE</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Systematic representation of EFSM model</td>
<td>20</td>
</tr>
<tr>
<td>4.1</td>
<td>Variable $signal_vector</td>
<td>30</td>
</tr>
<tr>
<td>4.2</td>
<td>EFSM model at state RST</td>
<td>31</td>
</tr>
<tr>
<td>4.3</td>
<td>Structure of variable $flag_complex</td>
<td>32</td>
</tr>
<tr>
<td>4.4</td>
<td>Complete $flag_complex value for DUT b04</td>
<td>33</td>
</tr>
<tr>
<td>4.5</td>
<td>Enable function</td>
<td>35</td>
</tr>
<tr>
<td>4.6</td>
<td>Update function</td>
<td>35</td>
</tr>
<tr>
<td>4.7</td>
<td>Example of constraint model</td>
<td>38</td>
</tr>
<tr>
<td>4.8</td>
<td>Fault free and faulty operation</td>
<td>39</td>
</tr>
<tr>
<td>5.1</td>
<td>VCS run result on b04</td>
<td>45</td>
</tr>
<tr>
<td>5.2</td>
<td>Summary of fault coverage and test generation time for b04 and b14</td>
<td>49</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>FIGURE NO.</th>
<th>TITLE</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>High-level overview of project implementation and scope</td>
<td>15</td>
</tr>
<tr>
<td>3.2</td>
<td>Work break down of design stages in sequence</td>
<td>15</td>
</tr>
<tr>
<td>3.3</td>
<td>High-level flowchart of project flow</td>
<td>16</td>
</tr>
<tr>
<td>3.4</td>
<td>State transition graph of a simple EFSM</td>
<td>18</td>
</tr>
<tr>
<td>3.5</td>
<td>Snippet of VHDL converted into EFSM</td>
<td>19</td>
</tr>
<tr>
<td>3.6</td>
<td>Snippet of VHDL converted into EFSM in Perl</td>
<td>20</td>
</tr>
<tr>
<td>3.7</td>
<td>Entity of SystemVerilog code</td>
<td>23</td>
</tr>
<tr>
<td>3.8</td>
<td>Overview of test pattern generation</td>
<td>24</td>
</tr>
<tr>
<td>4.1</td>
<td>Block diagram of DUT b04</td>
<td>26</td>
</tr>
<tr>
<td>4.2</td>
<td>High level overview of software design</td>
<td>28</td>
</tr>
<tr>
<td>4.3</td>
<td>Flowchart of <em>CM_GEN.pl</em></td>
<td>29</td>
</tr>
<tr>
<td>4.4</td>
<td>Typical VHDL design entity</td>
<td>30</td>
</tr>
<tr>
<td>4.5</td>
<td>EFSM model of DUT b04</td>
<td>32</td>
</tr>
<tr>
<td>4.6</td>
<td>EFSM model of DUT b04 with fault activation and propagation</td>
<td>37</td>
</tr>
<tr>
<td>4.7</td>
<td>Constraint models of fault free design</td>
<td>39</td>
</tr>
</tbody>
</table>
4.8 Constraint models of faulty design 40
4.9 Detection constraint 41
4.10 Program block 42
4.11 Output of VCS run 43
5.1 Test pattern from VCS run on b04 46
5.2 Simulation waveform of fault free b04 (addition at L6) 47
5.3 Simulation waveforms of faulty b04 (subtraction at L6) 47
5.4 Simulation waveforms of faulty b04 (multiplication at L6) 48
5.5 Simulation waveforms of faulty b04 (division at L6) 48
# LIST OF ABBREVIATIONS

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATPG</td>
<td>Automatic Test Pattern Generation</td>
</tr>
<tr>
<td>CMOS</td>
<td>Complementary Metal Oxide Semiconductor</td>
</tr>
<tr>
<td>CS</td>
<td>Constraint Sequence</td>
</tr>
<tr>
<td>DFT</td>
<td>Design For Test</td>
</tr>
<tr>
<td>DUT</td>
<td>Design-Under-Test</td>
</tr>
<tr>
<td>EDA</td>
<td>Electronic Design Automation</td>
</tr>
<tr>
<td>EFWM</td>
<td>Extended Finite State Machine</td>
</tr>
<tr>
<td>HDL</td>
<td>Hardware Description Language</td>
</tr>
<tr>
<td>HDVL</td>
<td>Hardware Description and Verification Language</td>
</tr>
<tr>
<td>IC</td>
<td>Integrated Circuit</td>
</tr>
<tr>
<td>ITC'99</td>
<td>1999 International Test Conference</td>
</tr>
<tr>
<td>LSA</td>
<td>Line-Stuck-At</td>
</tr>
<tr>
<td>LSI</td>
<td>Large Scale Integrated</td>
</tr>
<tr>
<td>VCS</td>
<td>Verilog Compiler Simulator</td>
</tr>
<tr>
<td>VHDL</td>
<td>VHSIC Hardware Description Language</td>
</tr>
<tr>
<td>VHSIC</td>
<td>Very-High-Speed Integrated Circuit</td>
</tr>
<tr>
<td>VLSI</td>
<td>Very-Large-Scale Integration</td>
</tr>
<tr>
<td>OUT</td>
<td>Operation-Under-Test</td>
</tr>
<tr>
<td>RTL</td>
<td>Register Transfer Level</td>
</tr>
<tr>
<td>SST</td>
<td>Single-State-Transition</td>
</tr>
</tbody>
</table>
# LIST OF APPENDICES

<table>
<thead>
<tr>
<th>APPENDIX</th>
<th>TITLE</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>SystemVerilog Test Bench Code for DUT b04</td>
<td>55</td>
</tr>
</tbody>
</table>
CHAPTER 1

INTRODUCTION

1.1 Project Background

Semiconductor technology has been improving rapidly lately and as it advances further into the nanometer regime, manufacturing processes become more defect-prone. Integrated circuit (IC) validation and testing play a very important role in ensuring and maintaining product quality while meeting the constraint of time-to-market. Ultimately the main goal is to obtain fault model that provides high fault coverage and requires short time for test generation.

Besides, the ever increasing complexity of digital designs are causing gate-level sequential test generation to become more time consuming and challenging. Many studies and researches have been done to achieve faster and better test sequence generation. Subsequently, there were some attempts to obtain higher fault coverage within reasonable time. One of them is the design-for-testability (DFT), such as scan methodology but this technique introduces additional hardware and causes area overhead.

Alternatively, we can perform test generation process at higher level of abstraction of the digital design, called functional test generation technique. In this technique, desired circuit functions are specified using hardware description languages (HDLs). The HDLs do all this without burdening the designer with the structural details of the circuit’s implementation.
Since HDLs are used to describe hardware at a high-level, by definition, the language constructs must be related to the actual hardware. This relationship has a higher degree of abstraction than the relationship between the gate-level representation of a design and the hardware. The test generation algorithms designed for the high-level models are usually direct extensions of those for the gate-level models, in which the functional modules are treated as primitive components and this allows improvement because fewer components are evaluated during test generation. The potential performance advantage of the reduced structural complexity makes this approach more attractive.

Since the conventional single LSA (Line Stuck-At) fault model is no longer suitable at this abstraction level, functional fault model is introduced to support this high level test generation platform. Currently, more complete functional fault models which can represent failure at more syntax of HDL description are Fummi’s fault model (Fummi et al., 1998) and Chen’s fault model (Chen, 2003). Fummi introduced a fault model that consists of bit failure and condition failure. Bit failure covers LSA of each bit of variable, signal or port, while condition failure covers LSA in each condition which may remove some execution paths in the erroneous HDL description.

Micro operation fault was introduced in Chen’s fault model, where it is a failure of micro operation to perform its intended function. The operators for the micro operation can be logical operators, relational operators, unary operators and arithmetic operators. An operator may fail to any other operator in its category. This fault is mapped by replacing the operator considered with its counter operator which must be defined. This project looks into Chen’s fault model for micro operation fault, specifically on arithmetic operators like addition, subtraction, multiplication and division.
1.2 Problem Statement

A better functional fault model called Enhanced Micro Operation Fault Model (Ooi and Fujiwara, 2010) was introduced to overcome Fummi’s and Chen’s fault models in the early 2010. In order to implement these fault models effectively without too much manual labor, an automated functional test generation platform is required especially when these fault model targets hard-to-test circuit. Unfortunately, currently only proprietary version of automation is developed (Chen and Noh, 1998) and it is not available for public usage.

Full automation is necessary in this context because in order to generate test pattern using micro operation fault model, it involves steps like conversion of behavioral HDL into Extended Finite State Machine (EFSM). Also, with automation, one can easily trigger intended fault, and find its state justification path and propagation path to build the constraint sequence. Besides that, the conventional method to build constraint model is using manual effort and the designer has to always refer back to the behavioral model of the circuit. All these can be eliminated with the automated test pattern generation platform.

1.3 Objectives

The main objective of this project is to build an automated constraint models generation platform via series of software programming which basically reads in the behavioral model of a design under test (DUT), generates the corresponding constraint sequence and constraint model. The constraint models generated here shall be useful for high-level test pattern generation. This project does not only focus on the implementation part, but also to be tested on dedicated validation benchmark circuits from the 1999 International Test Conference (ITC’99). This is to analyze its functionality and effectiveness in terms of fault coverage and test generation time.
1.4 Scopes of Study

This project involves a series of research and work on functional fault modeling, high level automatic test pattern generation and constraint solving in order to develop a software program that is capable of generating test pattern based on micro operation fault model. The scope of work begins with the understanding of behavioral model of DUT in VHDL (very-high-speed-integrated-circuit hardware description language). This is because the benchmark circuits from ITC’99 are all released in VHDL code. It is important to differentiate the design entity of the DUT which basically consists of entity declaration and architecture block. In VHDL behavioral modeling, the code contains sequential statements that are executed sequentially in a predefined order. This order of the statements in the code is important and may affect the semantics of the code.

EFSM was studied to understand all of its entities. EFSM plays a very important role to model the DUT from VHDL and to derive test sequences. These test sequences are derived such that all the transitions of the EFSM are traversed to guarantee that every statement in the behavioral description is verified. Constraint models using SystemVerilog are generated from these test sequences.

By using constraint solver tool called VCS (Verilog Compiler Simulator, from Synopsys) to solve the constraint models, the corresponding test pattern can be generated. These test patterns are used to activate and propagate targeted faults. Another software tool called Altera Quartus II is used to simulate the DUT with the test pattern obtained earlier. The simulation waveforms can be used to validate and verify the quality of the test patterns.

The main software program is written using Perl programming language. Perl was chosen because it has very strong regular expression functions and its multiple levels of hashes of hashes can be easily implemented as database to store all the information of the EFSM. Besides that, Perl is a good scripting language and can
be easily used to detect an arithmetic operation and trigger an intended fault to obtain the test sequence.

1.5 Organization of the Report

This report consists of six chapters with the brief description of each chapter as stated below:

Chapter 1 presents the introduction to this project, including the background of the project, problem statements, objectives of the project and scopes of the study.

Chapter 2 provides literature reviews from previous research work on high level ATPG and functional fault modeling.

Chapter 3 discusses the methodology applied in this project, including the project planning and schedule, design and implementation workflow, and software involved in this project.

Chapter 4 describes the software design and test bench design of the project. This chapter discuss about the architecture and structure of the main program and how SystemVerilog test bench is built as input for test pattern generation.

Chapter 5 discusses the result from the main software program, test bench building and VCS run. The verification of the run result is discussed as well, followed by analysis of the result.

Chapter 6 concludes the overall work done in this project. Limitations of the design in this project were discussed, including recommendation for future work.
REFERENCES


