FROM LOGIS GATE TO VERILOG CODE A HANDBOOK GUIDE

Authors:
Hasliza Hassan

email:
hliza@uthm.edu.my

Abstract:
"From Logic Gate to Verilog Code: A Handbook Guide" is an essential resource for anyone interested in understanding the fundamentals of digital design and programming using Verilog. This comprehensive handbook takes readers on a journey from the basics of logic gates and digital circuits to advanced Verilog coding techniques, making it accessible to beginners while also providing valuable insights for experienced engineers.

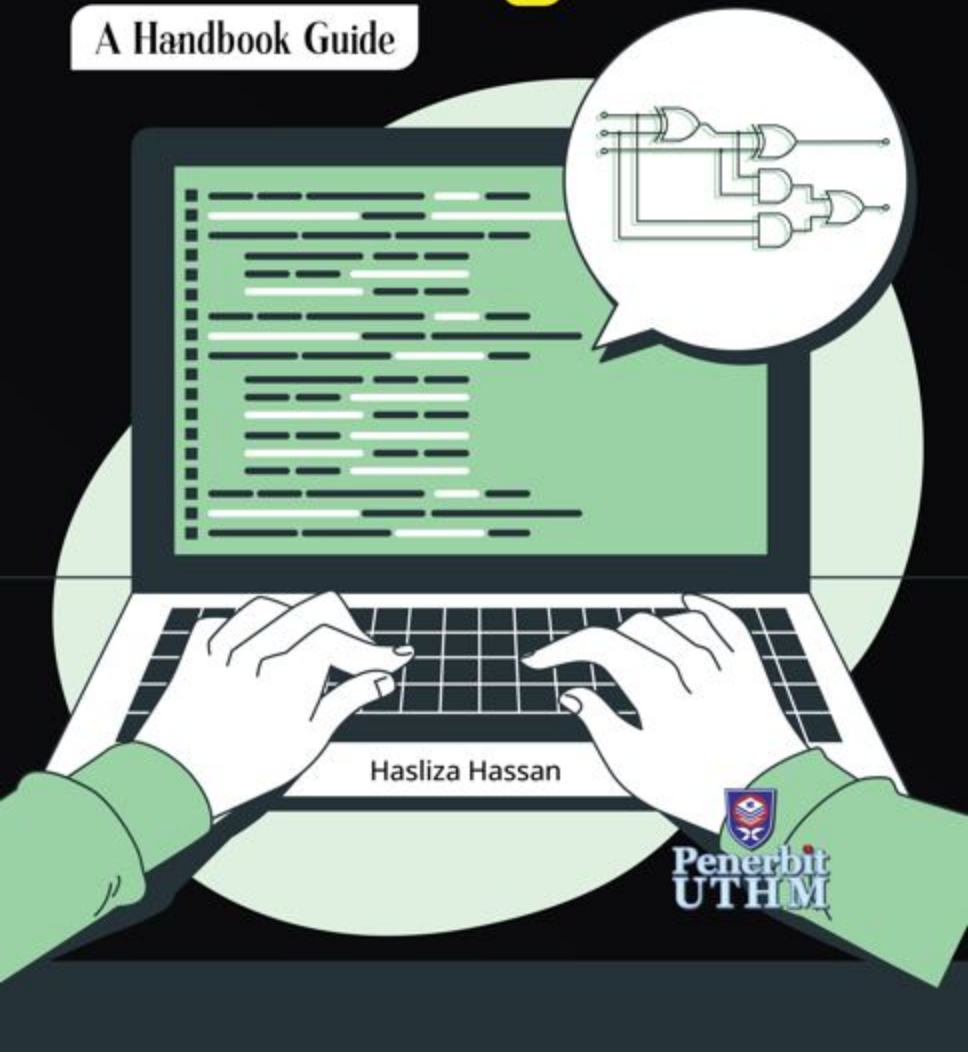
The book begins by introducing the foundational concepts of digital logic, including Boolean algebra, logic gates, and basic circuit design principles. It then progresses to explain the role of Verilog in digital design and covers topics such as module instantiation, data types, operators, and behavioural modelling.

One of the key strengths of this handbook is its hands- on approach. Each chapter is accompanied by practical examples and exercises that allow readers to apply their knowledge and reinforce their understanding of Verilog coding. The book also includes step-by-step guides for designing common digital circuits and implementing them in Verilog, making it a valuable reference for both learning and practical application.

Keywords: fundamentals of digital design and programming using Verilog, digital circuits to advanced Verilog coding techniques

From Logic Gate to Verilog Code

A Handbook Guide



Hasliza Hassan

Penerbit UTHM

First Published 2024

Authors: Hasliza Hassan

Published & Printed by:
Penerbit UTHM
Universiti Tun Hussein Onn Malaysia
86400 Parit Raja,
Batu Pahat, Johor
Tel: 07-453 8529/8698
Fax: 07-453 6145

Website: http://penerbit.uthm.edu.my
E-mail: pt@uthm.edu.my
http://e-bookstore.uthm.edu.my

Penerbit UTHM is a member of
Majlis Penerbitan Ilmiah Malaysia
(MAPIM)

Cataloguing-in-Publication Data

Perpustakaan Negara Malaysia

A catalogue record for this book is available from the National Library of Malaysia

ISBN 978-629-490-097-4

From Logic Gate to Verilog Code

A Handbook Guide

Hasliza Hassan

Penerbit UTHM
2024

TABLE OF CONTENT

List of Figures vii
List of Tables ix
Preface xi

CHAPTER 1
INTRODUCTION TO DIGITAL LOGIC AND DESIGN

1.1 Introduction 1
1.2 Significance of Digital Electronics in Modern Technology 1
1.3 Analog and Digital Signals 2
1.4 Number Systems and Binary Arithmetic 4
1.5 Logic Gates and Truth Tables 6
1.6 Boolean Algebra 9
1.7 Karnaugh Map (K-Map) 10
1.8 Combinational Logic Circuits 13

CHAPTER 2
SEQUENTIAL LOGIC AND FLIP-FLOPS

2.1 SR, JK, D, and T Flip-Flop 15
2.2 Design and Analysis of Sequential Circuits 23
2.3 Analysis of Sequential Circuits 26
2.4 Finite State Machines (FSMs) 27

CHAPTER 3
COMBINATIONAL CIRCUIT DESIGN

3.1 Combinational Digital Design 31
3.2 Multiplexers and Demultiplexers 32
3.3 Demultiplexers (DEMUX) 38

3.4 Encoders and Decoders 41
3.5 Adders and Subtractors 48

CHAPTER 4
INTRODUCTION TO VERILOG PROGRAMMING

4.1 Basics of Verilog 53
4.2 Verilog HDL Module 57
4.3 Verilog Conditional Operator 58
4.4 Verilog Procedural Statements 59
4.5 Register-transfer Level 62
4.6 Language Constructs and Conventions in Verilog 63

CHAPTER 5
COMPILATION OF CIRCUIT DESIGN TO VERILOG DESIGN

5.1 Gate Level Modelling 67
5.2 Behavioural Modelling 71
5.3 Dataflow Description 74
5.4 Testbench and Synthesis 78

Bibliography 81
Biography 83
Index 85

LIST OF FIGURES

Figure 1.1:	AND gate	6
Figure 1.2:	OR gate	6
Figure 1.3:	NOT gate	7
Figure 1.4:	NOT gate	7
Figure 1.5:	NOR gate	8
Figure 1.6:	XOR gate	8
Figure 1.7:	Basic component in combinational logic circuit	13
Figure 1.8:	Examples of combinational circuits	15
Figure 2.1:	SR flip-flop	18
Figure 2.2:	D flip-flop from SR flip-flop	19
Figure 2.3:	JK flip-flop	20
Figure 2.4:	Waveform JK flip-flop	21
Figure 2.5:	Symbol for JK flip-flop	21
Figure 2.6:	D flip-flop	22
Figure 2.7:	T flip-flop	23
Figure 2.8:	Example for sequential circuit using flip-flop	26
Figure 2.9:	Components of a finite state machine	28
Figure 3.1:	Circuit MUX 2 to 1	31
Figure 3.2:	Symbol 4x1 MUX	33
Figure 3.3:	Circuit 4x1 multiplexer in gate level	34
Figure 3.4:	Circuit 8x1 multiplexer using 4x1 multiplexer in gate level	35
Figure 3.5:	Circuit 16x1 multiplexer using 8x1 multiplexer in gate level	37
Figure 3.6:	Circuit 1x4 demultiplexer	39
Figure 3.7:	Circuit 1x8 demultiplexer	39
Figure 3.8:	Circuit 1x6 demultiplexer	41

PREFACE

This book is intended for students, engineers, and hobbyists who want to learn how to design digital systems using the hardware description language Verilog. Verilog is a powerful and versatile language that can be used to describe the structure and behaviour of digital circuits at various levels of abstraction, from gates and wires to complex components and systems. Verilog can also be used to simulate and test the functionality of digital designs before they are implemented in hardware. The main goal of this book is to provide a comprehensive guide to the process of transforming logic circuits into Verilog code. Logic circuits are graphical representations of digital systems that use symbols and connections to show how logic gates and other components are combined to perform a desired function. This book will teach It how to translate logic circuits into Verilog code using a systematic and step-by-step approach. It will learn how to identify the inputs, outputs, and internal signals of a logic circuit, how to choose the appropriate data types and operators for each signal, how to write the Verilog code that describes the structure and behavior of each component, and how to combine the components into a complete system.

Chapter 1
INTRODUCTION TO DIGITAL LOGIC AND DESIGN

1.1 INTRODUCTION

Digital electronics is a branch of electronics that deals with the use of discrete signals to represent and process information. In digital electronics, information is encoded as binary digits (0 and 1), known as bits, and is manipulated using logic gates and digital circuits. This approach allows for the precise and efficient handling of information and is fundamental to modern technology. Therefore, this book is a reference for digital electronics students who want to learn the basics of developing a system using Verilog code. This book is a fundamental book that uses basic knowledge in digital electronics. However, the reference and revision of the theoretical basis was carried out using the books at [1-10].

1.2 SIGNIFICANCE OF DIGITAL ELECTRONICS IN MODERN TECHNOLOGY

Information Processing : Digital electronics plays a crucial role in processing, storing, and transmitting vast amounts of information efficiently. It enables the representation of text, images, audio, and video in a digital format, making it easy to manipulate and transport data.

Computing : Digital electronics forms the foundation of modern computing devices, such as computers, tablets, and smartphones. It allows for complex calculations, data analysis, and software execution, enabling a wide range of applications from business to entertainment.

Communication : Digital electronics is at the heart of modern communication systems, including the internet, mobile networks, and satellite communications. Digital signals are robust against noise and can be transmitted over long distances without significant degradation.

Data Storage : Digital electronics enables the creation of high-capacity and reliable data storage devices, such as hard drives, solid-state drives, and optical discs. This is essential for preserving and accessing digital information.

Chapter 2

SEQUENTIAL LOGIC AND FLIP-FLOPS

2.1 SR, JK, D AND T FLIP-FLOP

Flip-flops are essential sequential logic circuits used in digital electronics to store and control binary data. They can hold one bit of information because they are bistable devices, which have two stable states. There are several types of flip-flops, such as JK, SR, D, and T flip-flops, each with its unique characteristics and applications:

2.1.1 SR Flip-Flop (Set-Reset Flip-Flop)

An SR flip-flop, also known as an SR latch (Set-Reset latch), is another type of digital sequential logic circuit that functions as a memory element in digital systems. Like the JK flip-flop, the SR flip-flop can store one bit of data (0 or 1) and can be used in various applications such as memory storage, data registers, and control circuits. The operation of an SR flip-flop:

i. Inputs

- S (Set): The S input sets the output Q to 1 (Q = 1) when S = 1.
- R (Reset): The R input resets the output Q to 0 (Q = 0) when R = 1.

ii. Outputs

- Q: Represents the output state of the flip-flop, which can be either 0 or 1.
- Q′ (Q complement): Represents the complementary output of Q, meaning Q′ is the inverse of Q (Q′ = 1 when Q = 0, and vice versa).

iii. Operation

- When both S and R inputs are 0, the SR flip-flop maintains its current state (Q and Q′ hold their values).
- When S = 1 and R = 0, the flip-flop sets its output Q to 1 (Q = 1) and Q′ to 0 (Q′ = 0), regardless of the previous state.

Chapter 3

COMBINATIONAL CIRCUIT DESIGN

3.1 COMBINATIONAL DIGITAL DESIGN

In contrast, sequential logic circuits contain some memory since their outputs are dependent on both their inputs at that particular time and their previous output state. Combinational logic circuits only produce outputs that are determined by the logic function of the current input state, which is either logic "0" or logic "1" at any given time. Because combinational logic circuits do not have feedback, any changes made to the signals supplied to their inputs will immediately impact the output. If not, a combinational logic circuit's output is always reliant on the arrangement of its inputs. This means that a combinational circuit has no memory.

Combinational logic circuits are constructed from more complex switching circuits by connecting fundamental logic gates such as NOR, NOT, or NAND. Combinational logic circuits can be extremely basic or extremely complex, and as NAND and NOR gates are regarded as "universal" gates, any combinational circuit can be created with just these two gates.

One type of combinational circuit is a decoder, which accepts binary input data and outputs several lines of code, each of which generates an equivalent decimal code at the output. A combinational logic circuit's function may be specified in three primary ways:

i. Boolean algebra: This creates the algebraic statement that illustrates how the logic circuit operates for each True or False input variable that yields a logic "1" output.

ii. Truth Table - A truth table provides a summary of all the output states for every conceivable combination of input variables that the gate may encounter in tabular form, therefore defining the function of a logic gate.

Chapter 4
INTRODUCTION TO VERILOG PROGRAMMING

4.1 BASICS OF VERILOG

Originally intended for use in the specification, documentation, and simulation of digital circuits to be built via VLSI processes, Verilog was developed as a language. The fact that it could be able to automatically generate the circuit from a description if it had one of sufficient quality was only discovered much later.

Electronic systems are modeled using Verilog, a hardware description language (HDL) that is standardized as IEEE 1364. It is frequently used in the register-transfer abstraction level of digital circuit design and verification. Verilog is also used in the design of genetic circuits, mixed-signal circuit verification, and analog circuit verification. Verilog's syntax is like the C programming language. It is case-sensitive and features basic preprocessor capabilities. It uses blocking (=) and non-blocking (<=) assignment operators. The non-blocking assignment allows for state-machine updates without temporary storage variables. Verilog requires explicit bit-width declarations for variables, unlike C where sizes are inferred from the variable type.

A hierarchy of modules makes up a Verilog design (Module-Based Design). Modules use input, output, and bidirectional ports to communicate with one another and to encapsulate design hierarchy. A module's internal structure may include instances of other modules (sub-hierarchies), concurrent and sequential statement blocks, and net/variable declarations. Verilog-AMS combines classic Verilog with analog and mixed-signal modeling. It makes an effort to close the gap between analog and digital realms.

The VLSI (Very Large Scale Integration) design flow is a systematic process that engineers and designers follow to create integrated circuits (ICs) or chips as shown in Figure 5.1. This process involves multiple stages, each with its own set of tasks and tools, to ensure the successful design and manufacturing of complex semiconductor devices.

Chapter 5

COMPILATION OF CIRCUIT DESIGN TO VERILOG DESIGN

5.1 GATE LEVEL MODELLING

Digital designers are normally familiar with all the common logic gates, their symbols, and their working. Flip-flops are built from the logic gates. All other functionally complex and more involved circuits can also be built using the basic gates. All the basic gates are available as "Primitives" in Verilog. Primitives are generalized modules that already exist in Verilog. They can be instantiated directly in other modules. Gate-level modelling in Verilog involves describing digital circuits using primitive logic gates, such as AND, OR, NOT, and others. It represents the design at a lower level of abstraction, making it easier to understand the hardware behaviour and optimize for performance and area.

5.1.1 Example of Gate-level Modeling

An example of a gate-level Verilog code for a simple AND gate

```
module and_gate(output reg y, input a, b);
 always @(a or b)
begin y = a & b;
end
endmodule
```

5.1.2 Steps for Gate-level Optimization

These steps is to optimize gate-level Verilog design for better performance:

i. Minimize logic depth: Reduce the number of gate levels to decrease propagation delay and improve performance.

ii. Use efficient logic gates: Choose the most appropriate gates (e.g., NAND, NOR) to optimize area and power consumption.

iii. Eliminate redundant logic: Remove unnecessary gates or logic to simplify the design and save resources.

BIBLIOGRAPHY

Advanced Digital Design with Verilog HDL - Michel D. Ciletti, PHI, 2009.

Advanced Digital Logic Design using Verilog, State Machines & Synthesis for FPGA - Sunggu Lee, Cengage Learning, 2012.

Combinational Logic Circuits. www.electronics-tutorials.ws. Retrieved March 10, 2024, from https://www.electronicstutorials.ws/combination/comb_1.html

Deschamps, Jean-Pierre, Valderrama, Elena, Terés, Lluis. Digital Systems. Springer, 1st Edition 2017.

DIGITAL DESIGN THROUGH VERILOG.pdf (mrcet.com)

Donzellini, G., Oneto, L., Ponta, D., Anguita, D. Introduction to Digital Systems Design. Springer, 1st Edition, 2019.

Floyd, Thomas L. Digital Fundamentals. Pearson Education International, 11th Edition, 2015.

Fundamentals of Digital Logic with Verilog Design - Stephen Brown,Zvonkoc Vranesic, TMH, 2nd Edition.

Gate-level Modeling and Optimization in Verilog (unrepo.com)

John F. Wakerly, Digital Design: Principles and Practices. Pearson Education International, 8th Edition, 2008.

Tocci, Ronald J., Moss, Gregory L. and Neal, S. Widmer. Digital Systems: Principles and Applications. Pearson Education International, 12th Edition, 2017.

Tokheim, Roger L. Digital Electronics: Principles and Applications, 8th Edition, 2008.

Zainalabdien Navabi, Verliog Digital System Design, TMH, 2nd Edition.

BIOGRAPHY

Hasliza Hassan is a lecturer in the Department of Electrical Engineering Technology, Faculty of Engineering Technology, UTHM. She obtained her Bachelor's Degree in Microelectronics Engineering from Universiti Malaysia Perlis (UniMAP) and went to pursue her Master's degree at Universiti Kebangsaan Malaysia (UKM). Then pursued her Doctor of Philosophy (PhD) at Universiti Teknologi Malaysia in Electrical Engineering. She is committed to contribute to the development of electronic engineering and strengthen the microelectronics industry through research and teaching. Apart from excellence in engineering, Hasliza Hassan is a passionate author in sharing life experiences and thoughts through various forms of media, including creative writing.

INDEX

A

algebra 8, 9, 10, 13, 27, 29
AND gate 36, 44, 45, 56, 65, 66

B

binary 1, 3, 4, 5, 6, 8, 10, 12, 15, 26, 29, 39, 40, 41, 42, 46, 47, 48, 62, 69
Binary Arithmetic 4

C

Combinational circuits 12
combinational logic 13, 27, 29, 39
combinational logic circuits 29, 39

D

data flow 72, 73, 74
decimal 4, 5, 29
De Morgan's 9, 13
Demultiplexers (DEMUX) 30
D flip-flop 17, 20

F

full adder 46, 73, 77

I

inverter 44, 45

J

JK flip-flop 15, 17, 18, 19

K

K-map 10, 11, 22, 23

L

logic gates 1, 12, 13, 27, 29, 43, 65

M

Multiplexers (MUX) 30

N

NAND gate 12, 16
NOR gate 16, 29

O

OR gate 14, 31, 40, 41, 44, 66

S

sequential circuit 20, 21, 24
SR flip-flop 15, 16, 17, 19
synchronous counter 22, 24

T

testbench 55, 67, 70, 71, 76, 77
T flip-flop 15, 21

X

XOR gate 72, 74

From Logic Gate to Verilog Code

A Handbook Guide

"From Logic Gate to Verilog Code: A Handbook Guide" is an essential resource for anyone interested in understanding the fundamentals of digital design and programming using Verilog. This comprehensive handbook takes readers on a journey from the basics of logic gates and digital circuits to advanced Verilog coding techniques, making it accessible to beginners while also providing valuable insights for experienced engineers.

The book begins by introducing the foundational concepts of digital logic, including Boolean algebra, logic gates, and basic circuit design principles. It then progresses to explain the role of Verilog in digital design and covers topics such as module instantiation, data types, operators, and behavioural modelling.

One of the key strengths of this handbook is its hands- on approach. Each chapter is accompanied by practical examples and exercises that allow readers to apply their knowledge and reinforce their understanding of Verilog coding. The book also includes step-by-step guides for designing common digital circuits and implementing them in Verilog, making it a valuable reference for both learning and practical application.

Penerbit UTHM

ISBN 978-629-490-097-4

03500

9 786294 900974

For more information, please scan the code