

1. Record Nr.	UNINA9910743250303321
Autore	Taraate Vaibbhav
Titolo	Digital logic design using Verilog : coding and RTL synthesis // Vaibbhav Taraate
Pubbl/distr/stampa	Singapore : , : Springer, , [2022] ©2022
ISBN	981-16-3199-9 981-16-3198-0
Edizione	[2nd ed.]
Descrizione fisica	1 online resource (607 pages)
Disciplina	371.320973
Soggetti	Logic design - Data processing Verilog (Computer hardware description language)
Lingua di pubblicazione	Inglese
Formato	Materiale a stampa
Livello bibliografico	Monografia
Nota di bibliografia	Includes bibliographical references and index.
Nota di contenuto	Intro -- Preface -- Acknowledgements -- Contents -- About the Author -- 1 Introduction -- 1.1 Evolution of Logic Design -- 1.2 System and Logic Design Abstractions -- 1.2.1 Architecture Design -- 1.2.2 Micro-architecture Design -- 1.2.3 RTL Design and Synthesis -- 1.2.4 Switch Level Design -- 1.3 Integrated Circuit Design and Methodologies -- 1.3.1 RTL Design -- 1.3.2 Functional Verification -- 1.3.3 Synthesis -- 1.3.4 Physical Design -- 1.4 Verilog as Hardware Description Language -- 1.5 Verilog Design Description -- 1.5.1 Structural Design -- 1.5.2 Behavior Design -- 1.5.3 Synthesizable Design -- 1.6 Few Important Verilog Terminologies -- 1.7 Exercises -- 1.8 Summary -- 2 Concept of Concurrency and Verilog Operators -- 2.1 Use of Continuous Assignment to Model Design -- 2.2 Use of always Procedural Block to Implement Combinational Design -- 2.3 Concept of Concurrency -- 2.4 Verilog Arithmetic Operators -- 2.5 Verilog Logical Operators -- 2.6 Verilog Equality and Inequality Operators -- 2.7 Verilog Sign Operators -- 2.8 Verilog Bitwise Operators -- 2.9 Verilog Relational Operators -- 2.10 Verilog Concatenation and Replication Operators -- 2.11 Verilog Reduction Operators -- 2.12 Verilog Shift Operators -- 2.13 Exercises -- 2.14 Summary -- 3 Verilog Constructs and Combinational Design-I -- 3.1 The Role of Constructs -- 3.2 Logic Gates and Synthesizable RTL --

3.2.1 NOT or Invert Logic -- 3.2.2 OR Logic -- 3.2.3 NOR Logic --  
3.2.4 AND Logic -- 3.2.5 NAND Logic -- 3.2.6 Two Input XOR Logic --  
3.2.7 Two Input XNOR Logic -- 3.3 Tristate Logic -- 3.4 Arithmetic  
Circuits -- 3.4.1 Adder -- 3.4.1.1 Half Adder -- 3.4.1.2 Full Adder --  
3.4.2 Subtractor -- 3.4.2.1 Half Subtractor -- 3.4.2.2 Full Subtractor --  
3.5 Exercises -- 3.6 Summary -- 4 Verilog Constructs and  
Combinational Design-II -- 4.1 Procedural Block always @\*.  
4.2 Multi-bit Adders and Subtractors -- 4.2.1 Four-Bit Full Adder --  
4.2.2 4-Bit Full Subtractor -- 4.2.3 4-Bit Adder and Subtractor -- 4.3  
Optimization of Resources -- 4.3.1 Optimization Using Only Adders --  
4.3.2 Optimization by Tweaking the Logic to Have Better Data and  
Control Path -- 4.4 Procedural Block initial -- 4.5 Simulation Concepts:  
Basic Testbench -- 4.6 Comparators and Parity Detectors -- 4.6.1  
Binary Comparators -- 4.6.2 Parity Detector -- 4.7 Code Converters --  
4.7.1 Binary to Gray Code Converter -- 4.7.2 Gray to Binary Code  
Converter -- 4.8 Let Us Think About the Design from Specifications --  
4.9 Exercises -- 4.10 Summary -- 5 Multiplexers as Universal Logic --  
5.1 Multiplexers -- 5.2 Multiplexer as Universal logic -- 5.2.1 2:1 MUX  
-- 5.3 The if...else Versus case Construct -- 5.4 The 4:1 MUX Using if...  
else -- 5.5 The 4:1 MUX Using case Construct -- 5.6 The 4:1 Mux  
Using 2:1 MUX -- 5.7 Let Us Design Combinational Logic Using  
Multiplexers -- 5.8 Optimization Strategies Using RTL Tweaks -- 5.9  
Exercises -- 5.10 Summary -- 6 Decoders and Encoders -- 6.1  
Decoders -- 6.1.1 1 Line to 2 Decoder Using case construct -- 6.1.2 1  
Line to 2 Decoder Having Enable Using case -- 6.1.3 2 Line to 4  
Decoder with Enable Using case -- 6.1.4 2 Line to 4 Decoder with  
Active Low Enable Using case -- 6.1.5 2 to 4 Decoder Using Continuous  
Assignments -- 6.1.6 Decoder Using Shift Operator -- 6.1.7 Testbench  
of 2:4 Decoder -- 6.1.8 4 Line to 16 Decoder Using 2:4 Decoder -- 6.2  
Testbench for 4:16 Decoder -- 6.3 Encoders -- 6.3.1 Priority Encoders  
-- 6.4 Testbench of 4:2 Priority encoder -- 6.5 Exercises -- 6.6  
Summary -- 7 Event Queue and Design Guidelines -- 7.1 Verilog  
Stratified Event Queue -- 7.2 Verilog Blocking Assignments -- 7.3  
Incomplete Sensitivity List -- 7.4 Continuous Versus Procedural  
Assignments -- 7.5 Combinational Loops in Design.  
7.6 Unintentional Latches in the Design -- 7.7 Use of Blocking  
Assignments -- 7.8 Use of if...else Versus case constructs -- 7.9  
Nested Multiplexer or Priority Logic -- 7.10 Parallel Logic or Decoding  
Logic -- 7.11 Priority Encoding Structure -- 7.12 Missing  
default Condition in case construct -- 7.13 Nested if...else with Missing  
else Condition -- 7.14 Logical Equality Versus Case Equality -- 7.14.1  
Logical Equality and Logical Inequality Operators -- 7.14.2 Case  
Equality and Case Inequality Operators -- 7.15 Multiple Driver  
Assignments -- 7.16 Exercises -- 7.17 Summary -- 8 Basics of  
Sequential Design Using Verilog -- 8.1 Sequential Logic -- 8.1.1  
Positive-Level Sensitive D Latch -- 8.1.2 Negative-Level Sensitive D  
Latch -- 8.2 Flip-Flop -- 8.2.1 Positive Edge-Triggered D Flip-Flop --  
8.2.2 Negative Edge-Triggered D Flip-Flop -- 8.2.3 Synchronous and  
Asynchronous Reset -- 8.2.3.1 D Flip-Flop Having Asynchronous Reset  
-- 8.2.4 D Flip-Flop Having Synchronous Reset -- 8.2.5 Flip-Flop  
Having Synchronous Load Enable and Asynchronous Reset -- 8.2.6  
Flip-Flop with Synchronous Load and Synchronous Reset -- 8.3  
Exercises -- 8.4 Summary -- 9 Synchronous Counter Design Using  
Synthesizable Constructs -- 9.1 Synchronous Counters -- 9.1.1 Three-  
Bit Up Counter -- 9.1.2 Three-Bit Down Counter -- 9.1.3 Three-Bit  
Up-Down Counter -- 9.2 Gray Counters -- 9.2.1 Gray and Binary  
Counter -- 9.2.2 Ring Counters -- 9.2.3 Johnson Counters -- 9.3 BCD  
Up-Down Counter -- 9.4 Exercises -- 9.5 Summary -- 10 RTL Design

of Registers and Memories -- 10.1 Parallel Input and Parallel Output (PIPO) Register -- 10.2 Shift Register -- 10.3 Right and Left Shift Operation -- 10.4 Timing and Performance Evaluation -- 10.5 Asynchronous Counter Design -- 10.5.1 Ripple Counters -- 10.6 RTL Design of Memories -- 10.7 Parameterized Read-Write Memory -- 10.8 Exercises -- 10.9 Summary.

11 Sequential Circuit Design Guidelines -- 11.1 What Happens If Blocking Assignments Are Used to Code Sequential Logic? -- 11.1.1 Blocking Assignments and Multiple always Blocks -- 11.1.2 Multiple Blocking Assignments Used in the Single always Block -- 11.1.3 Example Blocking Assignment -- 11.2 Non-blocking Assignments -- 11.2.1 Example Non-blocking Assignments -- 11.2.2 Example Non-blocking Assignment -- 11.2.3 Example Using Non-blocking Assignments -- 11.3 Latch Versus Flip-Flop -- 11.3.1 D Flip-Flop -- 11.3.2 Latch -- 11.4 Use of Synchronous Versus Asynchronous Reset -- 11.4.1 D Flip-Flop Having Asynchronous Reset -- 11.4.2 Synchronous Reset D Flip-Flop -- 11.5 Use of if...else Versus case constructs -- 11.6 Internally Generated Clocks -- 11.7 Guidelines for Modeling Synchronous Designs -- 11.8 Multiple Clocks in the Same module -- 11.9 Multi-phase Clocks in the Design -- 11.10 Guidelines for Modeling Asynchronous Designs -- 11.11 Exercises -- 11.12 Summary -- 12 RTL Design Strategies for Complex Designs -- 12.1 ALU Design -- 12.1.1 Logic Unit Design -- 12.1.1.1 Logic Unit to Infer Parallel Logic -- 12.1.1.2 Logic Unit Having Registered Inputs and Outputs -- 12.1.2 Arithmetic Unit -- 12.1.3 Arithmetic and Logic Unit -- 12.2 Functions and Tasks -- 12.2.1 Counting Number of 1's from the Given String -- 12.2.2 RTL Design Using function to Count Number of 1'S -- 12.3 Synthesis Result of RTL Using function -- 12.4 Synthesis Result of RTL Using task -- 12.5 Exercises -- 12.6 Summary -- 13 RTL Tweaks and Performance Improvement Techniques -- 13.1 Arithmetic Resource Sharing -- 13.1.1 RTL Design Using Resource Sharing to Have Area Optimization -- 13.2 Gated Clocks and Dynamic Power Reduction -- 13.3 Use of Pipelining in Design -- 13.3.1 Design Without Pipelining -- 13.3.2 Speed Improvement Using Register Balancing or Pipelining. -- 13.4 Counter Design and Duty Cycle Control -- 13.5 MOD-3 Counter RTL Design to Have 50% Duty Cycle -- 13.6 Exercise -- 13.7 Summary -- 14 Finite State Machines Using Verilog -- 14.1 Moore Versus Mealy Machines -- 14.1.1 Level to Pulse Converter -- 14.2 FSM Encoding Styles -- 14.2.1 Binary Encoding -- 14.2.1.1 Two-Bit Binary Counter FSM -- 14.2.2 Gray Encoding -- 14.2.2.1 Two-Bit Gray Counter FSM -- 14.3 One-Hot Encoding -- 14.4 Sequence Detectors Using FSMs -- 14.4.1 Mealy Sequence Detector Using Two always Procedural Blocks -- 14.4.2 Mealy Machine: Sequence Detector to Detect 101 Overlapping Sequence -- 14.5 Improving the Design Performance for FSMs -- 14.6 Exercises -- 14.7 Summary -- 15 Non-synthesizable Verilog Constructs and Testbenches -- 15.1 Intra-delay and Inter-delay Assignments -- 15.1.1 Simulation for Blocking Assignments -- 15.1.2 Simulation of Non-blocking Assignments -- 15.2 The always and initial Procedural Block -- 15.2.1 Blocking Assignments with Inter-assignment Delays -- 15.2.2 Blocking Assignments with Intra-assignment Delays -- 15.2.3 Non-blocking Assignments with Inter-assignment Delays -- 15.2.4 Non-blocking Assignments with Intra-assignment Delays -- 15.3 Role of Testbenches -- 15.4 Multiple Assignments Within the begin-end -- 15.5 Multiple Assignments Within the fork-join -- 15.6 Display Tasks -- 15.7 Exercises -- 15.8 Summary -- 16 FPGA Architecture and Design Flow -- 16.1 Introduction to PLD -- 16.2 FPGA as Programmable ASIC -- 16.2.1 SRAM Based FPGA -- 16.2.2 Flash Based FPGA -- 16.2.3 Antifuse

FPGAS -- 16.2.4 Important FPGA Blocks -- 16.3 FPGA Design Flow --  
16.3.1 Design Entry -- 16.3.2 Design Simulation and Synthesis --  
16.3.3 Design Implementation -- 16.3.4 Device Programming -- 16.4  
Logic Realization Using FPGA -- 16.4.1 Configurable Logic Block --  
16.4.2 Input Output Block (IOB) -- 16.4.3 Block RAM.  
16.4.4 Digital Clock Manager (DCM) Block.

---