Appendix A: List of Worked Examples

Example 2.1 Converting Decimal to Decimal 12
Example 2.2 Converting Binary to Decimal 13
Example 2.3 Converting Octal to Decimal 13
Example 2.4 Converting Hexadecimal to Decimal 14
Example 2.5 Converting Decimal to Binary 16
Example 2.6 Converting Decimal to Octal 17
Example 2.7 Converting Decimal to Hexadecimal 18
Example 2.8 Converting Binary to Octal 19
Example 2.9 Converting Binary to Hexadecimal 20
Example 2.10 Converting Octal to Binary 20
Example 2.11 Converting Hexadecimal to Binary 21
Example 2.12 Converting Octal to Hexadecimal 21
Example 2.13 Converting Hexadecimal to Octal 22
Example 2.14 Single-Bit Binary Addition 23
Example 2.15 Multiple-Bit Binary Addition 23
Example 2.16 Single-Bit Binary Subtraction 24
Example 2.17 Multiple-Bit Binary Subtraction 24
Example 2.18 Finding the Range of an Unsigned Number 26
Example 2.19 Decimal Values that a 4-Bit, Signed Magnitude Code Can Represent 27
Example 2.20 Finding the Range of a Signed Magnitude Number 28
Example 2.21 Finding the Decimal Value of a Signed Magnitude Number 28
Example 2.22 Decimal Values that a 4-Bit, One's Complement Code Can Represent 29
Example 2.23 Finding the Range of a 1's Complement Number 30
Example 2.24 Finding the Decimal Value of a 1's Complement Number 30
Example 2.25 Decimal Values that a 4-bit, Two's Complement Code can Represent 31
Example 2.26 Finding the Range of a Two's Complement Number 32
Example 2.27 Finding the Decimal Value of a Two's Complement Number 32
Example 2.28 Finding the Two's Complement Code of a Decimal Number 33
Example 2.29 Two's Complement Addition 35
Example 3.1 Calculating I_{CC} and I_{GND} when Sourcing Multiple Loads 55
Example 3.2 Calculating I_{CC} and I_{GND} when both Sourcing and Sinking Loads 56
Example 3.3 Determining if Specifications are Violated when Driving Another Gate as a Load 78
Example 3.4 Determining the Output Current when Driving Multiple Gates as the Load 79
Example 3.5 Determining the Output Current when Driving a Pull-Up Resistor as the Load 80
Example 3.6 Determining the Output Current when Driving a Pull-Down Resistor as the Load 81
Example 3.7 Determining the Output Current when Driving an LED where HIGH = ON 82
Example 3.8 Determining the Output Current when Driving an LED where HIGH = OFF 83
Example 4.1 Proving DeMorgan's Theorem of Duality Using Proof by Exhaustion 96
Example 4.2 Converting Between Positive and Negative Logic Using Duality 97
Example 4.3 Using the Commutative Property to Untangle Crossed Wires 101
Example 4.4 Using the Associative Property to Address Fan-In Limitations 102
Example 4.5 Using the Distributive Property to Reduce the Number of Logic Gates in a Circuit 103
Example 4.6 Proving the Absorption Theorem Using Proof by Exhaustion 104
Example 4.7 Proving of the Uniting Theorem 105
Example 4.8 Converting a Sum of Products form into One that Uses only NAND Gates 107
Example 4.9 Converting a Product of Sums form into One that Uses only NOR Gates 108
Example 4.10 Using DeMorgan's Theorem in Algebraic form (1) 109
Example 4.11 Using DeMorgan's Theorem in Algebraic form (2) 109
Example 4.12 Determining the Logic Expression from a Logic Diagram 112
Example 4.13 Determining the Truth Table from a Logic Diagram 113
Example 4.14 Determining the Delay of a Combinational Logic Circuit 114

© Springer Nature Switzerland AG 2019

B. J. LaMeres, *Introduction to Logic Circuits & Logic Design with Verilog*,
https://doi.org/10.1007/978-3-030-13605-5

Example 4.15 Creating a Canonical Sum of Products Logic Circuit Using Minterms 116
Example 4.16 Creating a Minterm List from a Truth Table 117
Example 4.17 Creating Equivalent Functional Representations from a Minterm List 118
Example 4.18 Creating a Product of Sums Logic Circuit using Maxterms 120
Example 4.19 Creating a Maxterm List from a Truth Table 121
Example 4.20 Creating Equivalent Functional Representations from a Maxterm List 122
Example 4.21 Creating Equivalent forms to Represent Logic Functionality 123
Example 4.22 Minimizing a Logic Expression Algebraically 125
Example 4.23 Using a K-map to Find a Minimized Sum of Products Expression (2-input) 130
Example 4.24 Using a K-map to Find a Minimized Sum of Products Expression (3-input) 131
Example 4.25 Using a K-map to Find a Minimized Sum of Products Expression (4-input) 132
Example 4.26 Using a K-map to Find a Minimized Product of Sums Expression (2-input) 133
Example 4.27 Using a K-map to Find a Minimized Product of Sums Expression (3-input) 134
Example 4.28 Using a K-map to Find a Minimized Product of Sums Expression (4-input) 135
Example 4.29 Deriving the Minimal Sum from a K-map 137
Example 4.30 Using Don't Cares to Produce a Minimal SOP Logic Expression 138
Example 4.31 Eliminating a Timing Hazard by Including Nonessential Product Terms 143
Example 5.1 Declaring Verilog Module Ports 169
Example 5.2 Modeling Combinational Logic using Continuous Assignment with Logical Operators 177
Example 5.3 Modeling Combinational Logic using Continuous Assignment with Conditional Operators (1) 178
Example 5.4 Modeling Combinational Logic using Continuous Assignment with Conditional Operators (2) 179
Example 5.5 Modeling Delay in Continuous Assignments 180
Example 5.6 Inertial Delay Modeling when using Continuous Assignment. 181
Example 5.7 Verilog Structural Design using Explicit Port Mapping 183
Example 5.8 Verilog Structural Design using Positional Port Mapping 184
Example 5.9 Modeling Combinational Logic Circuits Using Gate-Level Primitives 185
Example 5.10 Modeling Combinational Logic Circuits with a User-Defined Primitive 186
Example 6.1 2-to-4 One-Hot Decoder: logic synthesis by hand 196
Example 6.2 3-to-8 One-Hot Decoder: Verilog Modeling using Logical Operators 197
Example 6.3 3-to-8 One-Hot Decoder: Verilog Modeling using Conditional Operators 198
Example 6.4 7-Segment Display Decoder: Truth Table 199
Example 6.5 7-Segment Display Decoder: Logic Synthesis by Hand 200
Example 6.6 7-Segment Display Decoder: Verilog Modeling using Logical Operators 201
Example 6.7 7-Segment Display Decoder: Verilog Modeling using Conditional Operators 202
Example 6.8 4-to-2 binary Encoder: Logic Synthesis by Hand 203
Example 6.9 4-to-2 binary Encoder: Verilog Modeling using Logical and Conditional Operators 204
Example 6.10 2-to-1 Multiplexer: Logic Synthesis by Hand 205
Example 6.11 4-to-1 Multiplexer: Verilog Modeling using Logical and Conditional Operators 206
Example 6.12 1-to-2 Demultiplexer: Logic Synthesis by Hand 207
Example 6.13 1-to-4 Demultiplexer: Verilog Modeling using Logical and Conditional Operators 208
Example 7.1 Push-button Window Controller: Word Description 238
Example 7.2 Push-button Window Controller: State Diagram 240
Example 7.3 Push-button Window Controller: State Transition Table 241
Example 7.4 Solving for the Number of Bits Needed for Binary-State Encoding 243
Example 7.5 Push-Button Window Controller: State Encoding 245
Example 7.6 Push-Button Window Controller: Next State Logic 246
Example 7.7 Push-Button Window Controller: Output Logic 247
Example 7.8 Push-Button Window Controller: Logic Diagram 248
Example 7.9 Serial Bit Sequence Detector (part 1) 249
Example 7.10 Serial Bit Sequence Detector (part 2) 250
Example 7.11 Serial Bit Sequence Detector (part 3) 251
Example 7.12 Vending Machine Controller (part 1) 252
Example 7.13 Vending Machine Controller (part 2) 253
Example 7.14 Vending Machine Controller (part 3) 254

Example 7.15 2-Bit Binary Up Counter (part 1) ... 256
Example 7.16 2-Bit Binary Up Counter (part 2) ... 257
Example 7.17 2-Bit Binary Up/Down Counter (part 1) ... 258
Example 7.18 2-Bit Binary Up/Down Counter (part 2) ... 259
Example 7.19 2-Bit Gray Code Up Counter (part 1) ... 260
Example 7.20 2-Bit Gray Code Up Counter (part 2) ... 261
Example 7.21 2-Bit Gray Code Up/Down Counter (part 1) ... 262
Example 7.22 2-Bit Gray Code Up/Down Counter (part 2) ... 263
Example 7.23 3-Bit One-Hot Up Counter (part 1) ... 264
Example 7.24 3-Bit One-Hot Up Counter (part 2) ... 265
Example 7.25 3-Bit One-Hot Up/Down Counter (part 1) ... 266
Example 7.26 3-Bit one-Hot Up/Down Counter (part 2) ... 267
Example 7.27 3-Bit one-Hot Up/Down Counter (part 3) ... 268
Example 7.28 Determining the Next State Logic and Output Logic Expression of a FSM ... 271
Example 7.29 Determining the State Transition Table of a FSM ... 272
Example 7.30 Determining the State Diagram of a FSM ... 273
Example 7.31 Determining the Maximum Clock Frequency of a FSM ... 276
Example 8.1 Using Blocking Assignments to Model Combinational Logic ... 293
Example 8.2 Using Non-Blocking Assignments to Model Sequential Logic ... 293
Example 8.3 Identical Behavior when Using Blocking vs. Non-Blocking Assignments ... 294
Example 8.4 Different Behavior when Using Blocking vs. Non-Blocking Assignments (1) ... 295
Example 8.5 Different Behavior when using Blocking vs. Non-Blocking Assignments (2) ... 296
Example 8.6 Behavior of Statement Groups Begin/End vs. Fork/Join ... 297
Example 8.7 Using If-Else Statements to Model Combinational Logic ... 299
Example 8.8 Using Case Statements to Model Combinational Logic ... 301
Example 8.9 Test Bench for a Combinational Logic Circuit ... 310
Example 8.10 Test Bench for a Sequential Logic Circuit ... 311
Example 8.11 Printing Test Bench Results to the Transcript ... 312
Example 8.12 Test Bench with Automatic Output Checking ... 313
Example 8.13 Using a Loop to Generate Stimulus in a Test Bench ... 315
Example 8.14 Printing Test Bench Results to an External File ... 316
Example 8.15 Reading Test Bench Stimulus Vectors from an External File ... 317
Example 9.1 Behavioral Model of a D-Latch in Verilog ... 323
Example 9.2 Behavioral Model of a D-Flip-Flop in Verilog ... 324
Example 9.3 Behavioral Model of a D-Flip-Flop with Asynchronous Reset in Verilog ... 325
Example 9.4 Behavioral Model of a D-Flip-Flop with Asynchronous Reset and Preset in Verilog ... 326
Example 9.5 Behavioral Model of a D-Flip-Flop with Synchronous Enable in Verilog ... 327
Example 9.6 Push-Button Window Controller in Verilog: Design Description ... 328
Example 9.7 Push-Button Window Controller in Verilog: Port Definition ... 328
Example 9.8 Push-Button Window Controller in Verilog: Full Model ... 331
Example 9.9 Push-Button Window Controller in Verilog: Simulation Waveform ... 332
Example 9.10 Push-Button Window Controller in Verilog: Changing State Codes ... 333
Example 9.11 Serial Bit Sequence Detector in Verilog: Design Description and Port Definition ... 334
Example 9.12 Serial Bit Sequence Detector in Verilog: Full Model ... 335
Example 9.13 Serial Bit Sequence Detector in Verilog: Simulation Waveform ... 336
Example 9.14 Vending Machine Controller in Verilog: Design Description and Port Definition ... 336
Example 9.15 Vending Machine Controller in Verilog: Full Model ... 337
Example 9.16 Vending Machine Controller in Verilog: Simulation Waveform ... 338
Example 9.17 2-Bit Up/Down Counter in Verilog: Design Description and Port Definition ... 338
Example 9.18 2-Bit Up/Down Counter in Verilog: Full Model (Three-Block Approach) ... 339
Example 9.19 2-Bit Up/Down Counter in Verilog: Simulation Waveform ... 339
Example 9.20 Binary Counter using a Single-Procedural Block in Verilog ... 341
Example 9.21 Binary Counter with Range Checking in Verilog ... 342
Example 9.22 Binary Counter with Enable in Verilog ... 343
Example 9.23 Binary Counter with Load in Verilog ... 344
Example 9.24 RTL Model of an 8-Bit Register in Verilog ... 345
Example 9.25 Registers as Agents on a Data Bus: System Topology ... 346

Example 9.26 Registers as Agents on a Data Bus: RTL Model in Verilog 347
Example 9.27 Registers as Agents on a Data Bus: Simulation Waveform 347
Example 9.28 RTL Model of a 4-Stage, 8-Bit Shift Register in Verilog 348
Example 10.1 Calculating the Final Digit Line Voltage in a DRAM Based on Charge Sharing 372
Example 10.2 Behavioral Models of a 4 × 4 Asynchronous Read-only Memory in Verilog 376
Example 10.3 Behavioral Models of a 4 × 4 Synchronous Read-only Memory in Verilog 377
Example 10.4 Behavioral Model of a 4 × 4 Asynchronous Read/Write Memory in Verilog 378
Example 10.5 Behavioral Model of a 4 × 4 Synchronous Read/Write Memory in Verilog 379
Example 12.1 Design of a Half Adder 398
Example 12.2 Design of a Full Adder 398
Example 12.3 Design of a Full Adder Out of Half Adders 400
Example 12.4 Design of a 4-Bit Ripple Carry Adder (RCA) 401
Example 12.5 Timing Analysis of a 4-Bit Ripple Carry Adder 402
Example 12.6 Design of a 4-bit Carry Look Ahead Adder (CLA) – Overview 403
Example 12.7 Design of a 4-bit Carry Look Ahead Adder (CLA) – Algebraic Formation 404
Example 12.8 Timing Analysis of a 4-Bit Carry Look Ahead Adder 405
Example 12.9 Structural Model of a Full Adder Using Two Half Adders in Verilog 406
Example 12.10 Structural Model of a 4-Bit Ripple Carry Adder in Verilog 407
Example 12.11 Test Bench for a 4-Bit Ripple Carry Adder using Nested for Loops in Verilog 408
Example 12.12 Structural Model of a 4-bit Carry Look Ahead Adder in Verilog 409
Example 12.13 4-Bit Carry Look Ahead Adder – Simulation Waveform 409
Example 12.14 Behavioral Model of a 4-bit Adder in Verilog 410
Example 12.15 Design of a 4-Bit Subtractor Using Full Adders 411
Example 12.16 Creating a Programmable Inverter Using an XOR Gate 412
Example 12.17 Design of a 4-Bit Programmable Adder/Subtractor 412
Example 12.18 Performing Long Multiplication on Decimal Numbers 414
Example 12.19 Performing Long Multiplication on Binary Numbers 414
Example 12.20 Design of a Single-Bit Multiplier 415
Example 12.21 Design of a 4-Bit Unsigned Multiplier 415
Example 12.22 Timing Analysis of a 4-Bit Unsigned Multiplier 416
Example 12.23 Multiplying an Unsigned Binary Number by Two Using a Logical Shift Left 416
Example 12.24 Illustrating how an Unsigned Multiplier Incorrectly Handles Signed Numbers 417
Example 12.25 Process to Correctly Handle Signed Numbers Using an Unsigned Multiplier 418
Example 12.26 Performing Long Division on Decimal Numbers 419
Example 12.27 Performing Long Multiplication on Binary Numbers 420
Example 12.28 Design of a 4-Bit Unsigned Divider Using a Series of Iterative Subtractors 421
Example 12.29 Dividing 1111_2 (15_{10}) by 0111_2 (7_{10}) Using the Iterative Subtraction Architecture 422
Example 12.30 Dividing an Unsigned Binary Numbers by Two Using a Logical Shift Right 423
Example 13.1 Memory Map for a 256 × 8 Memory System 432
Example 13.2 Execution of an Instruction to “Load Register A Using Immediate Addressing” 435
Example 13.3 Execution of an Instruction to “Load Register A Using Direct Addressing” 436
Example 13.4 Execution of an Instruction to “Store Register A Using Direct Addressing” 437
Example 13.5 Execution of an Instruction to “Add Registers A and B” 438
Example 13.6 Execution of an Instruction to “Branch Always” 439
Example 13.7 Execution of an Instruction to “Branch if Equal to Zero” 440
Example 13.8 Top-Level Block Diagram for the 8-Bit Computer System 442
Example 13.9 Instruction Set for the 8-Bit Computer System 443
Example 13.10 Memory System Block Diagram for the 8-Bit Computer System 444
Example 13.11 CPU Block Diagram for the 8-Bit Computer System 448
Example 13.12 State Diagram for LDA_IMM 455
Example 13.13 Simulation Waveform for LDA_IMM 456
Example 13.14 State Diagram for LDA_DIR 457
Example 13.15 Simulation Waveform for LDA_DIR 458
Example 13.16 State Diagram for STA_DIR 459
Example 13.17 Simulation Waveform for STA_DIR 460
Example 13.18 State Diagram for ADD_AB 461
Example 13.19 Simulation Waveform for ADD_AB 462

Example 13.20 State Diagram for BRA 463
Example 13.21 Simulation Waveform for BRA 464
Example 13.22 State Diagram for BEQ 465
Example 13.23 Simulation Waveform for BEQ when Taking the Branch (Z = 1) 466
Example 13.24 Simulation Waveform for BEQ when the Branch is not Taken (Z = 0) 467

Appendix B: Concept Check Solutions

CC1.1	B
CC1.2	C
CC2.1	C
CC2.2	D
CC2.3	D
CC2.4	A
CC3.1	A
CC3.4(a)	B
CC3.4(b)	C
CC3.4(c)	A
CC3.3	A
CC3.4	D
CC4.1	B
CC4.2	B
CC4.3	D
CC4.4(a)	B
CC4.4(b)	A
CC4.5	D
CC5.1	D
CC5.2	C
CC5.3	A
CC5.4(a)	A
CC5.4(b)	B
CC5.5(a)	D
CC5.5(b)	A
CC5.6	B
CC5.7	C
CC6.1	C
CC6.2	D
CC6.3	C
CC6.4	C
CC7.1(a)	B
CC7.1(b)	D
CC7.2	A
CC7.3	C
CC7.4(a)	A
CC7.4(b)	C
CC7.4(c)	B
CC7.4(d)	D
CC7.4(e)	A
CC7.4(f)	C
CC7.5	A
CC7.6	D
CC7.7	B
CC8.1	A
CC8.2	B
CC8.3	B
CC8.4	B
CC9.1	D
CC9.2	D
CC9.3	C
CC9.4	A
CC9.5	C
CC10.1	D
CC10.2	C
CC10.3	B
CC10.4	A
CC11.1	C
CC11.2	B
CC12.1	B
CC12.2	D
CC12.3	B
CC12.4	A
CC13.1	B
CC13.2	D
CC13.3	D
CC13.4	B

© Springer Nature Switzerland AG 2019

B. J. LaMeres, *Introduction to Logic Circuits & Logic Design with Verilog*,
https://doi.org/10.1007/978-3-030-13605-5

Index

A

Absorption, 104
Abstraction, 157
AC specifications. *See* Switching characteristics
Adders
  in Verilog, 405
Adder/subtractor circuit, 411
Addition, 22, 397
AND gate, 46
Anti-fuse, 361
Associative property, 102
Asynchronous memory, 359
Axioms, 94
  logical negation, 94
  logical precedence, 95
  logical product, 94
  logical sum, 94
  logical values, 94

B

Base, 7
Base conversions, 11
  binary to decimal, 12
  binary to hexadecimal, 19
  binary to octal, 19
  decimal to binary, 15
  decimal to decimal, 11
  decimal to hexadecimal, 18
  decimal to octal, 17
  hexadecimal to binary, 21
  hexadecimal to decimal, 14
  hexadecimal to octal, 22
  octal to binary, 20
  octal to decimal, 13
  octal to hexadecimal, 21
Binary addition. *See* Addition
Binary number system, 9
Binary subtraction. *See* Subtraction
Bipolar junction transistor (BJT), 71
Bistable, 214
Boolean algebra, 93
Boolean algebra theorems, 95
Borrows, 24
Break-before-make switch behavior, 234
Buffer, 45
Byte, 10

C

Canonical product of sums, 118
Canonical sum of products, 115
Capacity, 355
Carry, 22
Carry look ahead adders (CLA), 402
Charge sharing, 371
Classical digital design flow, 161
CMOS. *See* Complementary metal oxide semiconductor (CMOS)
CMOS gates, 64
  inverter, 64
  NAND gate, 65
  NOR gate, 68
CMOS operation, 63
Combinational logic analysis, 111
Combining, 105
Commutative property, 100
Complementary metal oxide semiconductor (CMOS), 4, 62
Complements, 99
Complete sum, 137
Complex programmable logic device (CPLD), 387
Computer system design, 427
  addressing modes, 433
  arithmetic logic unit (ALU), 429
  central processing unit, 429
  condition code register, 429
  control unit, 429
  data memory, 428
  data path, 429
  direct addressing, 434
  example 8-bit system, 441
    control unit, 452
    CPU, 448
    detailed instruction execution, 454
    instruction set, 443
    memory system, 444
  general purpose registers, 429
  hardware, 427
  immediate addressing, 433
  indexed addressing, 434
  inherent addressing, 434
  input output ports, 428
  instruction register, 429
  instructions, 427
    branches, 438
    data manipulations, 437
    loads and stores, 434
  memory address register, 429
  memory-mapped, 431
  memory mapped system, 430
  opcodes, 433
  operands, 433
  program, 427

© Springer Nature Switzerland AG 2019
B. J. LaMeres, *Introduction to Logic Circuits & Logic Design with Verilog*,
https://doi.org/10.1007/978-3-030-13605-5

Computer system design (*cont.*)
    program counter, 429
    program memory, 428
    registers, 429
    software, 427, 433
Configurable logic block (CLB), 389
Conjunction ($\wedge$), 94
Converting between bases. *See* Base conversions
Converting between positive and negative logic, 96
Counters, 256
    designing by hand, 256
Covering, 104
Cross-coupled inverter pair, 213

D

Data sheet, 57
7400 DC operating conditions, 74
DC specifications, 52
    $I_{IH\text{-}max}$, 53
    $I_{IL\text{-}max}$, 53
    $I_{I\text{-}max}$, 53
    $I_{OH\text{-}max}$, 52
    $I_{OL\text{-}max}$, 52
    $I_{O\text{-}max}$, 52
    I_q (quiescent current), 54
    NM_H, 53
    NM_L, 53
    $V_{IH\text{-}max}$, 53
    $V_{IH\text{-}min}$, 53
    $V_{IL\text{-}max}$, 53
    $V_{IL\text{-}min}$, 53
    $V_{OH\text{-}max}$, 52
    $V_{OH\text{-}min}$, 52
    $V_{OL\text{-}max}$, 52
    $V_{OL\text{-}min}$, 52
Decimal number system, 9
Decoders, 195
De Morgan's Theorem of Duality, 95
De Morgan's theorems, 106
Demultiplexer design by hand, 207
Demultiplexers, 207
Design abstraction, 157
Design domains, 159
    behavioral domain, 159
    physical domain, 159
    structural domain, 159
Design levels, 159
    algorithmic level, 159
    circuit level, 159
    gate-level, 159
    register transfer level, 159
    system level, 159
Design simplicity, 4
D-flip-flop, 225
Digit, 9
Digital design flow, 161
Digit notation, 9
Diodes, 81
7400 DIP pinout, 75
Discrete components, 62
Disjunction ($\vee$), 94
Distinguished one cells, 137
Distributive property, 103
Division, 419
    by powers of 2, 422
    signed, 423
    unsigned, 419
    using iterative subtractions, 420
D latch, 224
Don't cares (X), 138
Double pole, double throw (DPDT) switch, 233
Double pole, single throw (DPST) switch, 233
Driving loads, 77
    driving LEDs, 82
    driving resistive loads, 79
Dual in-line package (DIP), 75
Duality, 95
Dynamic hazard, 142
Dynamic random-access memory (DRAM), 369

E

Electrically erasable programmable read-only memory (EEPROM), 364
Electrical signaling, 1
Encoders, 202
Erasable programmable read-only memory (EPROM), 362
Essential prime implicant, 137

F

Field-programmable gate array (FPGA), 388
Finite state machines (FSM), 238
    binary state encoding, 242
    design examples by hand, 249
    design process, 248
    final logic diagram, 247
    gray code state encoding, 243
    introduction, 238
    next state logic, 245
    one-hot state encoding, 244
    output logic, 246
    reset condition, 269
    state diagram, 239
    state memory, 242
    state transition table, 240
    state variables, 245
    synthesis by hand, 241
FLASH memory, 365
    NAND-FLASH, 365
    NOR-FLASH, 365

Floating-gate transistor, 362
Forward current (I_F), 81
Forward voltage (V_F), 81
Full adders, 398
Functionally complete sets, 110
Fuse, 361

G

Gajski and Kuhn's Y-chart, 159
Gates, 43
Generic array logic (GAL), 385
Glitches, 141

H

Half adders, 398
Hard array logic (HAL), 386
Hazards, 141
Hexadecimal number system, 10
History of HDLs, 154

I

Idempotent, 99
Identity theorem, 98
Input/output blocks (IOBs), 393
Integrated circuit, 62
Inverter, 46
Involution, 100

K

Karnaugh map (K-map), 125

L

Large scale integrated circuit (LSI) logic, 195
Leading zero, 9
Least significant bit (LSB), 10
Light emitting diodes (LEDs), 81
Logic block (LE), 389
Logic expression, 44
Logic families, 62
Logic function, 44
Logic HIGH, 51
Logic levels, 51
Logic LOW, 51
Logic minimization, 124
Logic symbol, 43
Logic synthesis, 115
Logic value, 51
Logic waveform, 45
Look-up table (LUT), 389

M

Mask read-only memory (MROM), 360
Maxterm list (Π), 120
Maxterms, 118
Mealy machine, 239
Medium scale integrated circuit (MSI) logic, 195
Memory map model, 355
Metal oxide semiconductor field effect transistor
  (MOSFET), 62
Metastability, 214
Minimal sum, 136, 137
Minimization, 124
Minimization of logic algebraically, 124
Minimization of logic using K-maps, 128
Minterm list ($\sum$), 117
Minterms, 115
Modern digital design flow, 161
Moore machine, 239
MOSFET. *See* Metal oxide semiconductor field effect
  transistor (MOSFET)
Most significant bit (MSB), 10
Multiplexer design by hand, 204
Multiplexer modeling in Verilog, 205
Multiplexers, 204
Multiplication, 413
  combinational multiplier, 415
  by powers of 2, 416
  shift and add approach, 413
  signed, 417
  unsigned, 413

N

NAND-debounce circuit, 235
NAND gate, 47
Negation ($\neg$), 94
Negative logic, 51
Nibble, 10
Noise, 3
Noise margin HIGH (MN_H), 53
Noise margin LOW (MN_L), 53
Non-volatile memory, 356
NOR gate, 47
NPN, 71
N-type MOSFET (NMOS), 63
Null element, 98
Numerals, 7

O

Octal number system, 10
Ohm's law, 79
One-hot binary encoder design by hand, 203
One-hot binary encoder modeling in Verilog, 203
One-hot decoder design by hand, 196
One-hot decoder modeling in Verilog, 196
One's complement numbers, 29
OR gate, 47
Output DC specifications. *See* DC specifications
Output logic macrocell (OLMC), 385

P

7400 Part numbering scheme, 73
Place and route, 161
PNP, 71
Positional number system, 7
Positional weight, 11
Positive logic, 51
Postulates, 94
Power consumption, 4
Power supplies, 54
  I_{CC}, 54
  I_{GND}, 54
  V_{CC}, 54
Prime implicant, 129
Product of sums (POS) form, 106
Programmable array logic (PAL), 384
Programmable interconnect points (PIPs), 392
Programmable logic array (PLA), 383
Programmable read-only memory (PROM), 361
Proof by exhaustion, 95
P-type MOSFET (PMOS), 63
Pull-down network, 64
Pull-up network, 64

Q

Quiescent current (I_q), 54

R

Radix, 7
Radix point, 8
Random-access-memory (RAM), 356
Range
  one's1 complement numbers, 29
  signed magnitude numbers, 27
  two's complement numbers, 31
  unsigned numbers, 25, 27, 29, 31
Read cycle, 355
Read-only memory (ROM), 356, 357
Read/write (RW) memory, 356
Ripple carry adders (RCA), 400
Ripple counter, 231

S

7-segment decoder design by hand, 198
7-segment decoder modeling in Verilog, 201
Semiconductor memory, 355
Sequential access memory, 356
Sequential logic analysis, 270
Sequential logic timing, 229
7400 series logic families, 73
Shift register, 237
Signaling, 1
Signed magnitude numbers, 26
Signed numbers, 26
Simple programmable logic device (SPLD), 387
Single pole, double throw (SPDT) switch, 233
Single pole, single throw (SPST) switch, 233
Sinking current, 52, 53
Small scale integrated circuit (SSI) logic, 195
Sourcing and sinking multiple loads, 56
Sourcing current, 52
Sourcing multiple loads, 55
SR latch, 217, 219
SR latch with enable, 222
Static 0 hazard, 142
Static 1 hazard, 142
Static random-access memory (SRAM), 366
Subtraction, 24, 411
Sum of products (SOP) form, 106
Switch debouncing, 232
Switching characteristics, 57
  t_f (fall time), 57
  t_{PHL} (propagation delay HIGH to LOW), 57
  t_{PLH} (propagation delay LOW to HIGH), 57
  t_r (rise time), 57
  t_t (transition time), 57
Synchronous memory, 359

T

Technology mapping, 161
Timing hazards, 141
Toggle flop (T-flop), 230
Trailing zero, 9
Transistor-transistor logic (TTL), 71
Transmitter/receiver circuit, 50
Truth table formation, 44
TTL operation, 71
Two's complement arithmetic, 34
Two's complement numbers, 31

U

Uniting, 105
Unsigned numbers, 25

V

Verification, 159
Verilog, 153
  always blocks, 290
  arrays, 167
  behavioral modeling techniques
    adders, 405
    agents on a bus, 346
    counters, 340
    up counter, 340
    up counters with enables, 342
    up counters with loads, 343
    up counter with range checking, 341
    D-flip-flops, 324
    D-flip-flop with enable, 326
    D-flip-flop with preset, 325
    D-flip-flop with reset, 324

D-latches, 323
finite state machines, 328
encoding styles, 332
next state logic, 329
output logic, 330
state memory, 329
state variables, 329
registers, 345
RTL modeling, 344
shift registers, 348
casex statements, 301
casez statements, 301
compiler directives, 171
include, 171
timescale, 171
continuous assignment, 176
continuous assignment with conditional operators, 177
continuous assignment with delay, 179
continuous assignment with logical operators, 176
counters, 340
data types, 165
disable, 304
drive strength, 165
finite state machines, 328
forever loops, 302
for loops, 303
gate level primitives, 184
history, 155
if-else statements, 298, 300
initial blocks, 290
net data types, 166
number formatting
binary, 167
decimal, 167
hex, 167
octal, 167
operators, 171
assignment, 171
bitwise logical, 171
bitwise replication, 174
Boolean logic, 172
concatenation, 173
conditional, 173
numerical, 174
precedence, 175
reduction, 172
relational, 172
parameters, 170
procedural assignment, 289
procedural blocks, 289
repeat loops, 303
resolution, 165
RTL modeling, 344
sensitivity lists, 291
signal declaration, 169
statement groups, 297
structural design and hierarchy, 182
explicity port mapping, 183
gate level primities, 184
instantiation, 182
positional port mapping, 183
user defined primities, 185
system tasks, 304
file I/O, 306
simulation control, 308
text I/O, 304
user defined primities, 185
value set, 165
variable data types, 166
vectors, 166
while loops, 302
Very large scale integrated circuit (VLSI) logic, 195
Volatile memory, 356

W

Weight, 11
Word, 10
Write cycle, 355

X

X-don't cares, 138
XNOR gate, 49
XOR gate, 48
XOR/XNOR gates in K-maps, 138

Y

Y-chart, 159