

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	15
EXAMPLE 2.6 CONVERTING DECIMAL TO OCTAL	16
EXAMPLE 2.7 CONVERTING DECIMAL TO HEXADECIMAL	17
EXAMPLE 2.8 CONVERTING BINARY TO OCTAL	18
EXAMPLE 2.9 CONVERTING BINARY TO HEXADECIMAL	18
EXAMPLE 2.10 CONVERTING OCTAL TO BINARY	19
EXAMPLE 2.11 CONVERTING HEXADECIMAL TO BINARY	19
EXAMPLE 2.12 CONVERTING OCTAL TO HEXADECIMAL	20
EXAMPLE 2.13 CONVERTING HEXADECIMAL TO OCTAL	20
EXAMPLE 2.14 SINGLE BIT BINARY ADDITION	21
EXAMPLE 2.15 MULTIPLE BIT BINARY ADDITION	22
EXAMPLE 2.16 SINGLE BIT BINARY SUBTRACTION	22
EXAMPLE 2.17 MULTIPLE BIT BINARY SUBTRACTION	23
EXAMPLE 2.18 FINDING THE RANGE OF AN UNSIGNED NUMBER	24
EXAMPLE 2.19 DECIMAL VALUES THAT A 4-BIT, SIGNED MAGNITUDE CODE CAN REPRESENT	25
EXAMPLE 2.20 FINDING THE RANGE OF A SIGNED MAGNITUDE NUMBER	26
EXAMPLE 2.21 FINDING THE DECIMAL VALUE OF A SIGNED MAGNITUDE NUMBER	26
EXAMPLE 2.22 DECIMAL VALUES THAT A 4-BIT, ONE'S COMPLEMENT CODE CAN REPRESENT	27
EXAMPLE 2.23 FINDING THE RANGE OF A 1'S COMPLEMENT NUMBER	28
EXAMPLE 2.24 FINDING THE DECIMAL VALUE OF A 1'S COMPLEMENT NUMBER	28
EXAMPLE 2.25 DECIMAL VALUES THAT A 4-BIT, TWO'S COMPLEMENT CODE CAN REPRESENT	29
EXAMPLE 2.26 FINDING THE RANGE OF A TWO'S COMPLEMENT NUMBER	30
EXAMPLE 2.27 FINDING THE DECIMAL VALUE OF A TWO'S COMPLEMENT NUMBER	30
EXAMPLE 2.28 FINDING THE TWO'S COMPLEMENT CODE OF A DECIMAL NUMBER	31
EXAMPLE 2.29 TWO'S COMPLEMENT ADDITION	32
EXAMPLE 3.1 CALCULATING I_{CC} AND I_{GND} WHEN SOURCING MULTIPLE LOADS	49
EXAMPLE 3.2 CALCULATING I_{CC} AND I_{GND} WHEN BOTH SOURCING AND SINKING LOADS	50
EXAMPLE 3.3 DETERMINING IF SPECIFICATIONS ARE VIOLATED WHEN DRIVING ANOTHER GATE AS A LOAD	72
EXAMPLE 3.4 DETERMINING THE OUTPUT CURRENT WHEN DRIVING MULTIPLE GATES AS THE LOAD	73
EXAMPLE 3.5 DETERMINING THE OUTPUT CURRENT WHEN DRIVING A PULL-UP RESISTOR AS THE LOAD	74
EXAMPLE 3.6 DETERMINING THE OUTPUT CURRENT WHEN DRIVING A PULL-DOWN RESISTOR AS THE LOAD	75
EXAMPLE 3.7 DETERMINING THE OUTPUT CURRENT WHEN DRIVING AN LED WHERE HIGH = ON	76
EXAMPLE 3.8 DETERMINING THE OUTPUT CURRENT WHEN DRIVING AN LED WHERE HIGH = OFF	77
EXAMPLE 4.1 PROVING DeMORGAN'S THEOREM OF DUALITY USING PROOF BY EXHAUSTION	84
EXAMPLE 4.2 CONVERTING BETWEEN POSITIVE AND NEGATIVE LOGIC USING DUALITY	85
EXAMPLE 4.3 USING THE COMMUTATIVE PROPERTY TO UNTANGLE CROSSED WIRES	89
EXAMPLE 4.4 USING THE ASSOCIATIVE PROPERTY TO ADDRESS FAN-IN LIMITATIONS	90
EXAMPLE 4.5 USING THE DISTRIBUTIVE PROPERTY TO REDUCE THE NUMBER OF LOGIC GATES IN A CIRCUIT	91
EXAMPLE 4.6 PROVING THE ABSORPTION THEOREM USING PROOF BY EXHAUSTION	92
EXAMPLE 4.7 PROVING OF THE UNITING THEOREM	93
EXAMPLE 4.8 CONVERTING A SUM OF PRODUCTS FORM INTO ONE THAT USES ONLY NAND GATES	95
EXAMPLE 4.9 CONVERTING A PRODUCT OF SUMS FORM INTO ONE THAT USES ONLY NOR GATES	96
EXAMPLE 4.10 USING DeMORGAN'S THEOREM IN ALGEBRAIC FORM (1)	97
EXAMPLE 4.11 USING DeMORGAN'S THEOREM IN ALGEBRAIC FORM (2)	97
EXAMPLE 4.12 DETERMINING THE LOGIC EXPRESSION FROM A LOGIC DIAGRAM	100
EXAMPLE 4.13 DETERMINING THE TRUTH TABLE FROM A LOGIC DIAGRAM	101
EXAMPLE 4.14 DETERMINING THE DELAY OF A COMBINATIONAL LOGIC CIRCUIT	102

EXAMPLE 4.15 CREATING A CANONICAL SUM OF PRODUCTS LOGIC CIRCUIT USING MINTERMS	104
EXAMPLE 4.16 CREATING A MINTERM LIST FROM A TRUTH TABLE	105
EXAMPLE 4.17 CREATING EQUIVALENT FUNCTIONAL REPRESENTATIONS FROM A MINTERM LIST	106
EXAMPLE 4.18 CREATING A PRODUCT OF SUMS LOGIC CIRCUIT USING MAXTERMS	108
EXAMPLE 4.19 CREATING A MAXTERM LIST FROM A TRUTH TABLE	109
EXAMPLE 4.20 CREATING EQUIVALENT FUNCTIONAL REPRESENTATIONS FROM A MAXTERM LIST	110
EXAMPLE 4.21 CREATING EQUIVALENT FORMS TO REPRESENT LOGIC FUNCTIONALITY	111
EXAMPLE 4.22 MINIMIZING A LOGIC EXPRESSION ALGEBRAICALLY	113
EXAMPLE 4.23 USING A K-MAP TO FIND A MINIMIZED SUM OF PRODUCTS EXPRESSION (2-INPUT)	118
EXAMPLE 4.24 USING A K-MAP TO FIND A MINIMIZED SUM OF PRODUCTS EXPRESSION (3-INPUT)	119
EXAMPLE 4.25 USING A K-MAP TO FIND A MINIMIZED SUM OF PRODUCTS EXPRESSION (4-INPUT)	120
EXAMPLE 4.26 USING A K-MAP TO FIND A MINIMIZED PRODUCT OF SUMS EXPRESSION (2-INPUT)	121
EXAMPLE 4.27 USING A K-MAP TO FIND A MINIMIZED PRODUCT OF SUMS EXPRESSION (3-INPUT)	122
EXAMPLE 4.28 USING A K-MAP TO FIND A MINIMIZED PRODUCT OF SUMS EXPRESSION (4-INPUT)	123
EXAMPLE 4.29 DERIVING THE MINIMAL SUM FROM A K-MAP	125
EXAMPLE 4.30 USING DON'T CARES TO PRODUCE A MINIMAL SOP LOGIC EXPRESSION	126
EXAMPLE 4.31 ELIMINATING A TIMING HAZARD BY INCLUDING NON-ESSENTIAL PRODUCT TERMS	131
EXAMPLE 5.1 DECLARING VERILOG MODULE PORTS	157
EXAMPLE 5.2 MODELING COMBINATIONAL LOGIC USING CONTINUOUS ASSIGNMENT WITH LOGICAL OPERATORS	165
EXAMPLE 5.3 MODELING COMBINATIONAL LOGIC USING CONTINUOUS ASSIGNMENT WITH CONDITIONAL OPERATORS (1)	166
EXAMPLE 5.4 MODELING COMBINATIONAL LOGIC USING CONTINUOUS ASSIGNMENT WITH CONDITIONAL OPERATORS (2)	167
EXAMPLE 5.5 MODELING DELAY IN CONTINUOUS ASSIGNMENTS	168
EXAMPLE 5.6 INERTIAL DELAY MODELING WHEN USING CONTINUOUS ASSIGNMENT.	169
EXAMPLE 5.7 VERILOG STRUCTURAL DESIGN USING EXPLICIT PORT MAPPING	171
EXAMPLE 5.8 VERILOG STRUCTURAL DESIGN USING POSITIONAL PORT MAPPING	172
EXAMPLE 5.9 MODELING COMBINATIONAL LOGIC CIRCUITS USING GATE LEVEL PRIMITIVES	173
EXAMPLE 5.10 MODELING COMBINATIONAL LOGIC CIRCUITS WITH A USER-DEFINED PRIMITIVE	174
EXAMPLE 6.1 2-TO-4 ONE-HOT DECODER – LOGIC DYNTHESIS BY HAND	182
EXAMPLE 6.2 3-TO-8 ONE-HOT DECODER – VERILOG MODELING USING LOGICAL OPERATORS	183
EXAMPLE 6.3 3-TO-8 ONE-HOT DECODER – VERILOG MODELING USING CONDITIONAL OPERATORS	184
EXAMPLE 6.4 7-SEGMENT DISPLAY DECODER – TRUTH TABLE	185
EXAMPLE 6.5 7-SEGMENT DISPLAY DECODER – LOGIC SYNTHESIS BY HAND	186
EXAMPLE 6.6 7-SEGMENT DISPLAY DECODER – VERILOG MODELING USING LOGICAL OPERATORS	187
EXAMPLE 6.7 7-SEGMENT DISPLAY DECODER – VERILOG MODELING USING CONDITIONAL OPERATORS	187
EXAMPLE 6.8 4-TO-2 BINARY ENCODER – LOGIC SYNTHESIS BY HAND	189
EXAMPLE 6.9 4-TO-2 BINARY ENCODER – VERILOG MODELING USING LOGICAL AND CONDITIONAL OPERATORS	190
EXAMPLE 6.10 2-TO-1 MULTIPLEXER – LOGIC SYNTHESIS BY HAND	191
EXAMPLE 6.11 4-TO-1 MULTIPLEXER – VERILOG MODELING USING LOGICAL AND CONDITIONAL OPERATORS	192
EXAMPLE 6.12 1-TO-2 DEMULTIPLEXER – LOGIC SYNTHESIS BY HAND	193
EXAMPLE 6.13 1-TO-4 DEMULTIPLEXER – VERILOG MODELING USING LOGICAL AND CONDITIONAL OPERATORS	194
EXAMPLE 7.1 PUSH-BUTTON WINDOW CONTROLLER – WORD DESCRIPTION	223
EXAMPLE 7.2 PUSH-BUTTON WINDOW CONTROLLER – STATE DIAGRAM	224
EXAMPLE 7.3 PUSH-BUTTON WINDOW CONTROLLER – STATE TRANSITION TABLE	225
EXAMPLE 7.4 SOLVING FOR THE NUMBER OF BITS NEEDED FOR BINARY STATE ENCODING	227
EXAMPLE 7.5 PUSH-BUTTON WINDOW CONTROLLER – STATE ENCODING	229
EXAMPLE 7.6 PUSH-BUTTON WINDOW CONTROLLER – NEXT STATE LOGIC	230
EXAMPLE 7.7 PUSH-BUTTON WINDOW CONTROLLER – OUTPUT LOGIC	231
EXAMPLE 7.8 PUSH-BUTTON WINDOW CONTROLLER – LOGIC DIAGRAM	232
EXAMPLE 7.9 SERIAL BIT SEQUENCE DETECTOR (PART 1)	234
EXAMPLE 7.10 SERIAL BIT SEQUENCE DETECTOR (PART 2)	235
EXAMPLE 7.11 SERIAL BIT SEQUENCE DETECTOR (PART 3)	236
EXAMPLE 7.12 VENDING MACHINE CONTROLLER (PART 1)	237
EXAMPLE 7.13 VENDING MACHINE CONTROLLER (PART 2)	238
EXAMPLE 7.14 VENDING MACHINE CONTROLLER (PART 3)	239
EXAMPLE 7.15 2-BIT BINARY UP COUNTER (PART 1)	241

EXAMPLE 7.16 2-BIT BINARY UP COUNTER (PART 2)	242
EXAMPLE 7.17 2-BIT BINARY UP/DOWN COUNTER (PART 1)	243
EXAMPLE 7.18 2-BIT BINARY UP/DOWN COUNTER (PART 2)	244
EXAMPLE 7.19 2-BIT GRAY CODE UP COUNTER (PART 1)	245
EXAMPLE 7.20 2-BIT GRAY CODE UP COUNTER (PART 2)	246
EXAMPLE 7.21 2-BIT GRAY CODE UP/DOWN COUNTER (PART 1)	247
EXAMPLE 7.22 2-BIT GRAY CODE UP/DOWN COUNTER (PART 2)	248
EXAMPLE 7.23 3-BIT ONE-HOT UP COUNTER (PART 1)	249
EXAMPLE 7.24 3-BIT ONE-HOT UP COUNTER (PART 2)	250
EXAMPLE 7.25 3-BIT ONE-HOT UP/DOWN COUNTER (PART 1)	251
EXAMPLE 7.26 3-BIT ONE-HOT UP/DOWN COUNTER (PART 2)	252
EXAMPLE 7.27 3-BIT ONE-HOT UP/DOWN COUNTER (PART 3)	253
EXAMPLE 7.28 DETERMINING THE NEXT STATE LOGIC AND OUTPUT LOGIC EXPRESSION OF A FSM	256
EXAMPLE 7.29 DETERMINING THE STATE TRANSITION TABLE OF A FSM	257
EXAMPLE 7.30 DETERMINING THE STATE DIAGRAM OF A FSM	258
EXAMPLE 7.31 DETERMINING THE MAXIMUM CLOCK FREQUENCY OF A FSM	261
EXAMPLE 8.1 USING BLOCKING ASSIGNMENTS TO MODEL COMBINATIONAL LOGIC	275
EXAMPLE 8.2 USING NON-BLOCKING ASSIGNMENTS TO MODEL SEQUENTIAL LOGIC	275
EXAMPLE 8.3 IDENTICAL BEHAVIOR WHEN USING BLOCKING VS. NON-BLOCKING ASSIGNMENTS	276
EXAMPLE 8.4 DIFFERENT BEHAVIOR WHEN USING BLOCKING VS. NON-BLOCKING ASSIGNMENTS (1)	277
EXAMPLE 8.5 DIFFERENT BEHAVIOR WHEN USING BLOCKING VS. NON-BLOCKING ASSIGNMENTS (2)	278
EXAMPLE 8.6 BEHAVIOR OF STATEMENT GROUPS BEGIN/END VS. FORK/JOIN	279
EXAMPLE 8.7 USING IF-ELSE STATEMENTS TO MODEL COMBINATIONAL LOGIC	281
EXAMPLE 8.8 USING CASE STATEMENTS TO MODEL COMBINATIONAL LOGIC	282
EXAMPLE 8.9 TEST BENCH FOR A COMBINATIONAL LOGIC CIRCUIT	291
EXAMPLE 8.10 TEST BENCH FOR A SEQUENTIAL LOGIC CIRCUIT	292
EXAMPLE 8.11 PRINTING TEST BENCH RESULTS TO THE TRANSCRIPT	293
EXAMPLE 8.12 TEST BENCH WITH AUTOMATIC OUTPUT CHECKING	294
EXAMPLE 8.13 USING A LOOP TO GENERATE STIMULUS IN A TEST BENCH	295
EXAMPLE 8.14 PRINTING TEST BENCH RESULTS TO AN EXTERNAL FILE	296
EXAMPLE 8.15 READING TEST BENCH STIMULUS VECTORS FROM AN EXTERNAL FILE	297
EXAMPLE 9.1 BEHAVIORAL MODEL OF A D-LATCH IN VERILOG	303
EXAMPLE 9.2 BEHAVIORAL MODEL OF A D-FLIP-FLOP IN VERILOG	304
EXAMPLE 9.3 BEHAVIORAL MODEL OF A D-FLIP-FLOP WITH ASYNCHRONOUS RESET IN VERILOG	305
EXAMPLE 9.4 BEHAVIORAL MODEL OF A D-FLIP-FLOP WITH ASYNCHRONOUS RESET AND PRESET IN VERILOG	306
EXAMPLE 9.5 BEHAVIORAL MODEL OF A D-FLIP-FLOP WITH SYNCHRONOUS ENABLE IN VERILOG	307
EXAMPLE 9.6 PUSH-BUTTON WINDOW CONTROLLER IN VERILOG – DESIGN DESCRIPTION	308
EXAMPLE 9.7 PUSH-BUTTON WINDOW CONTROLLER IN VERILOG – PORT DEFINITION	308
EXAMPLE 9.8 PUSH-BUTTON WINDOW CONTROLLER IN VERILOG – FULL MODEL	311
EXAMPLE 9.9 PUSH-BUTTON WINDOW CONTROLLER IN VERILOG – SIMULATION WAVEFORM	312
EXAMPLE 9.10 PUSH-BUTTON WINDOW CONTROLLER IN VERILOG – CHANGING STATE CODES	312
EXAMPLE 9.11 SERIAL BIT SEQUENCE DETECTOR IN VERILOG – DESIGN DESCRIPTION AND PORT DEFINITION	313
EXAMPLE 9.12 SERIAL BIT SEQUENCE DETECTOR IN VERILOG – FULL MODEL	314
EXAMPLE 9.13 SERIAL BIT SEQUENCE DETECTOR IN VERILOG – SIMULATION WAVEFORM	315
EXAMPLE 9.14 VENDING MACHINE CONTROLLER IN VERILOG – DESIGN DESCRIPTION AND PORT DEFINITION	315
EXAMPLE 9.15 VENDING MACHINE CONTROLLER IN VERILOG – FULL MODEL	316
EXAMPLE 9.16 VENDING MACHINE CONTROLLER IN VERILOG – SIMULATION WAVEFORM	317
EXAMPLE 9.17 2-BIT UP/DOWN COUNTER IN VERILOG – DESIGN DESCRIPTION AND PORT DEFINITION	317
EXAMPLE 9.18 2-BIT UP/DOWN COUNTER IN VERILOG – FULL MODEL (THREE BLOCK APPROACH)	318
EXAMPLE 9.19 2-BIT UP/DOWN COUNTER IN VERILOG – SIMULATION WAVEFORM	318
EXAMPLE 9.20 BINARY COUNTER USING A SINGLE PROCEDURAL BLOCK IN VERILOG	319
EXAMPLE 9.21 BINARY COUNTER WITH RANGE CHECKING IN VERILOG	320
EXAMPLE 9.22 BINARY COUNTER WITH ENABLE IN VERILOG	321
EXAMPLE 9.23 BINARY COUNTER WITH LOAD IN VERILOG	322
EXAMPLE 9.24 RTL MODEL OF AN 8-BIT REGISTER IN VERILOG	323

EXAMPLE 9.25 REGISTERS AS AGENTS ON A DATA BUS — SYSTEM TOPOLOGY	324
EXAMPLE 9.26 REGISTERS AS AGENTS ON A DATA BUS — RTL MODEL IN VERILOG	324
EXAMPLE 9.27 REGISTERS AS AGENTS ON A DATA BUS — SIMULATION WAVEFORM	325
EXAMPLE 9.28 RTL MODEL OF A 4-STAGE, 8-BIT SHIFT REGISTER IN VERILOG	326
EXAMPLE 10.1 CALCULATING THE FINAL DIGIT LINE VOLTAGE IN A DRAM BASED ON CHARGE SHARING	348
EXAMPLE 10.2 BEHAVIORAL MODELS OF A 4×4 ASYNCHRONOUS READ ONLY MEMORY IN VERILOG	352
EXAMPLE 10.3 BEHAVIORAL MODELS OF A 4×4 SYNCHRONOUS READ ONLY MEMORY IN VERILOG	353
EXAMPLE 10.4 BEHAVIORAL MODEL OF A 4×4 ASYNCHRONOUS READ/WRITE MEMORY IN VERILOG	354
EXAMPLE 10.5 BEHAVIORAL MODEL OF A 4×4 SYNCHRONOUS READ/WRITE MEMORY IN VERILOG	355
EXAMPLE 12.1 DESIGN OF A HALF ADDER	374
EXAMPLE 12.2 DESIGN OF A FULL ADDER	374
EXAMPLE 12.3 DESIGN OF A FULL ADDER OUT OF HALF ADDERS	376
EXAMPLE 12.4 DESIGN OF A 4-BIT RIPPLE CARRY ADDER (RCA)	377
EXAMPLE 12.5 TIMING ANALYSIS OF A 4-BIT RIPPLE CARRY ADDER	378
EXAMPLE 12.6 DESIGN OF A 4-BIT CARRY LOOK AHEAD ADDER (CLA) — OVERVIEW	379
EXAMPLE 12.7 DESIGN OF A 4-BIT CARRY LOOK AHEAD ADDER (CLA) — ALGEBRAIC FORMATION	380
EXAMPLE 12.8 TIMING ANALYSIS OF A 4-BIT CARRY LOOK AHEAD ADDER	381
EXAMPLE 12.9 STRUCTURAL MODEL OF A FULL ADDER USING TWO HALF ADDERS IN VERILOG	382
EXAMPLE 12.10 STRUCTURAL MODEL OF A 4-BIT RIPPLE CARRY ADDER IN VERILOG	383
EXAMPLE 12.11 TEST BENCH FOR A 4-BIT RIPPLE CARRY ADDER USING NESTED FOR LOOPS IN VERILOG	384
EXAMPLE 12.12 STRUCTURAL MODEL OF A 4-BIT CARRY LOOK AHEAD ADDER IN VERILOG	385
EXAMPLE 12.13 4-BIT CARRY LOOK AHEAD ADDER — SIMULATION WAVEFORM	385
EXAMPLE 12.14 BEHAVIORAL MODEL OF A 4-BIT ADDER IN VERILOG	386
EXAMPLE 12.15 DESIGN OF A 4-BIT SUBTRACTOR USING FULL ADDERS	387
EXAMPLE 12.16 CREATING A PROGRAMMABLE INVERTER USING AN XOR GATE	387
EXAMPLE 12.17 DESIGN OF A 4-BIT PROGRAMMABLE ADDER/SUBTRACTOR	388
EXAMPLE 12.18 PERFORMING LONG MULTIPLICATION ON DECIMAL NUMBERS	389
EXAMPLE 12.19 PERFORMING LONG MULTIPLICATION ON BINARY NUMBERS	390
EXAMPLE 12.20 DESIGN OF A SINGLE-BIT MULTIPLIER	390
EXAMPLE 12.21 DESIGN OF A 4-BIT UNSIGNED MULTIPLIER	391
EXAMPLE 12.22 TIMING ANALYSIS OF A 4-BIT UNSIGNED MULTIPLIER	392
EXAMPLE 12.23 MULTIPLYING AN UNSIGNED BINARY NUMBER BY TWO USING A LOGICAL SHIFT LEFT	392
EXAMPLE 12.24 ILLUSTRATING HOW AN UNSIGNED MULTIPLIER INCORRECTLY HANDLES SIGNED NUMBERS	393
EXAMPLE 12.25 PROCESS TO CORRECTLY HANDLE SIGNED NUMBERS USING AN UNSIGNED MULTIPLIER	394
EXAMPLE 12.26 PERFORMING LONG DIVISION ON DECIMAL NUMBERS	395
EXAMPLE 12.27 PERFORMING LONG MULTIPLICATION ON BINARY NUMBERS	396
EXAMPLE 12.28 DESIGN OF A 4-BIT UNSIGNED DIVIDER USING A SERIES OF ITERATIVE SUBTRACTORS	397
EXAMPLE 12.29 DIVIDING 1111_2 (15_{10}) BY 0111_2 (7_{10}) USING THE ITERATIVE SUBTRACTION ARCHITECTURE	398
EXAMPLE 12.30 DIVIDING AN UNSIGNED BINARY NUMBERS BY TWO USING A LOGICAL SHIFT RIGHT	399
EXAMPLE 13.1 MEMORY MAP FOR A 256×8 MEMORY SYSTEM	408
EXAMPLE 13.2 EXECUTION OF AN INSTRUCTION TO “LOAD REGISTER A USING IMMEDIATE ADDRESSING”	411
EXAMPLE 13.3 EXECUTION OF AN INSTRUCTION TO “LOAD REGISTER A USING DIRECT ADDRESSING”	412
EXAMPLE 13.4 EXECUTION OF AN INSTRUCTION TO “STORE REGISTER A USING DIRECT ADDRESSING”	413
EXAMPLE 13.5 EXECUTION OF AN INSTRUCTION TO “ADD REGISTERS A AND B”	414
EXAMPLE 13.6 EXECUTION OF AN INSTRUCTION TO “BRANCH ALWAYS”	415
EXAMPLE 13.7 EXECUTION OF AN INSTRUCTION TO “BRANCH IF EQUAL TO ZERO”	416
EXAMPLE 13.8 TOP LEVEL BLOCK DIAGRAM FOR THE 8-BIT COMPUTER SYSTEM	418
EXAMPLE 13.9 INSTRUCTION SET FOR THE 8-BIT COMPUTER SYSTEM	419
EXAMPLE 13.10 MEMORY SYSTEM BLOCK DIAGRAM FOR THE 8-BIT COMPUTER SYSTEM	420
EXAMPLE 13.11 CPU BLOCK DIAGRAM FOR THE 8-BIT COMPUTER SYSTEM	424
EXAMPLE 13.12 STATE DIAGRAM FOR LDA_IMM	431
EXAMPLE 13.13 SIMULATION WAVEFORM FOR LDA_IMM	432
EXAMPLE 13.14 STATE DIAGRAM FOR LDA_DIR	433
EXAMPLE 13.15 SIMULATION WAVEFORM FOR LDA_DIR	434
EXAMPLE 13.16 STATE DIAGRAM FOR STA_DIR	435

EXAMPLE 13.17 SIMULATION WAVEFORM FOR STA_DIR	436
EXAMPLE 13.18 STATE DIAGRAM FOR ADD_AB	437
EXAMPLE 13.19 SIMULATION WAVEFORM FOR ADD_AB	438
EXAMPLE 13.20 STATE DIAGRAM FOR BRA	439
EXAMPLE 13.21 SIMULATION WAVEFORM FOR BRA	440
EXAMPLE 13.22 STATE DIAGRAM FOR BEQ	441
EXAMPLE 13.23 SIMULATION WAVEFORM FOR BEQ WHEN TAKING THE BRANCH ($Z = 1$)	442
EXAMPLE 13.24 SIMULATION WAVEFORM FOR BEQ WHEN THE BRANCH IS NOT TAKEN ($Z = 0$)	443

Index

A

Absorption, 92
Abstraction, 145
AC specifications. *See* Switching characteristics
Adder/subtractor circuit, 387
Adders
 in Verilog, 381
Addition, 21, 373
AND gate, 40
Anti-fuse, 337
Associative property, 89
Asynchronous memory, 335
Axioms, 82
 logical negation, 82
 logical precedence, 83
 logical product, 82
 logical sum, 83
 logical values, 82

B

Base, 7
Base conversions, 11
 binary to decimal, 12
 binary to hexadecimal, 18
 binary to octal, 17
 decimal to binary, 15
 decimal to decimal, 11
 decimal to hexadecimal, 16
 decimal to octal, 15
 hexadecimal to binary, 19
 hexadecimal to decimal, 14
 hexadecimal to octal, 20
 octal to binary, 19
 octal to decimal, 13
 octal to hexadecimal, 19
Binary addition. *See* Addition
Binary number system, 9
Binary subtraction. *See* Subtraction
Bipolar junction transistor (BJT), 65
Bistable, 200
Boolean algebra, 81
Boolean algebra theorems, 83
Borrows, 22
Break-before-make switch behavior, 219
Buffer, 39
Byte, 10

C

Canonical product of sums, 106
Canonical sum of products, 103

Capacity, 331
Carry, 21
Carry look ahead adders (CLA), 378
Charge sharing, 347
Classical digital design flow, 149
CMOS. *See* Complementary metal oxide semiconductor (CMOS)
Combinational logic analysis, 99
Combining, 93
Commutative property, 88
Complementary metal oxide semiconductor (CMOS), 4, 56
 gates, 58
 CMOS inverter, 58
 CMOS NAND gate, 59
 CMOS NOR gate, 62
 operation, 57
Complements, 87
Complete sum, 125
Complex programmable logic device (CPLD), 363
Computer system design, 403, 410, 413, 414, 418, 419, 424, 427, 430
 addressing modes, 409
 arithmetic logic unit (ALU), 405
 central processing unit, 405
 condition code register, 405
 control unit, 405
 data memory, 404
 data path, 405
 direct addressing, 410
 example 8-bit system, 417
 control unit, 427
 CPU, 424
 detailed instruction execution, 430
 instruction set, 418
 memory system, 419
 general purpose registers, 405
 hardware, 403
 immediate addressing, 410
 indexed addressing, 410
 inherent addressing, 410
 input output ports, 404
 instructions, 403
 branches, 414
 data manipulations, 413
 loads and stores, 410
 register, 405
 memory address register, 405
 memory map, 408
 memory mapped system, 406
 opcodes, 409
 operands, 409

Computer system design (*cont.*)

- program, 403
 - counter, 405
 - memory, 404
- registers, 405
- software, 403, 409

Configurable logic block (CLB), 365

Conjunction (\wedge), 82

Converting between bases. See Base conversions

Converting between positive and negative logic, 84

Counters, 241

- designing by hand, 241

Covering, 92

Cross-coupled inverter pair, 199

D

Data sheet, 51

7400 DC operating conditions, 69

DC specifications, 46

- I_{L-max} , 47
- I_{IH-max} , 47
- I_{IL-max} , 47
- I_{O-max} , 46
- I_{OH-max} , 46
- I_{OL-max} , 46
- I_q (quiescent current), 48
- NM_H , 47
- NM_L , 47
- V_{IH-max} , 47
- V_{IH-min} , 47
- V_{IL-max} , 47
- V_{IL-min} , 47
- V_{OH-max} , 46
- V_{OH-min} , 46
- V_{OL-max} , 46
- V_{OL-min} , 46

Decimal number system, 9

Decoders, 181

DeMorgan's theorem, 93

DeMorgan's theorem of duality, 83

Demultiplexer design by hand, 193

Demultiplexers, 193

Design abstraction, 145

Design domains, 146

- behavioral, 146
- physical, 146
- structural, 146

Design levels, 146

- algorithmic, 146
- circuit, 146
- gate, 146
- register transfer, 146
- system, 146

Design simplicity, 3

D-flip-flop, 211

Digit, 9

- notation, 9

Digital design flow, 149

Diodes, 75

7400 DIP pin-out, 69

Discrete components, 56

Disjunction (\vee), 82

Distinguished one cells, 125

Distributive property, 91

Division, 395

- by powers of 2, 398
- signed, 399
- unsigned, 395
- using iterative subtractions, 396

D latch, 210

Don't cares (X), 125

Double pole, double throw (DPDT) switch, 218

Double pole, single throw (DPST) switch, 218

Driving loads, 71

- LEDs, 76
- resistive loads, 73

Dual in-line package (DIP), 69

Duality, 83

Dynamic hazard, 130

Dynamic random access memory (DRAM), 345

E

Electrical signaling, 1

Electrically erasable programmable read only memory (EEPROM), 340

Encoders, 188

Erasable programmable read only memory (EPROM), 338

Essential prime implicant, 125

F

Field programmable gate array (FPGA), 364

Finite state machines (FSM), 223

- binary state encoding, 226
- design examples by hand, 233
- design process, 232
- final logic diagram, 231
- gray code state encoding, 227
- introduction, 223
- next state logic, 229
- one-hot state encoding, 228
- output logic, 230
- reset condition, 254
- state diagram, 223
- state memory, 226
- state transition table, 225
- state variables, 229
- synthesis by hand, 225

FLASH memory, 341

- NAND-FLASH, 341

- NOR-FLASH, 341

Floating-gate transistor, 338

Forward current (I_F), 75

Forward voltage (V_F), 75
 Full adders, 374
 Functionally complete sets, 98
 Fuse, 337

G

Gajski and Kuhn's Y-chart, 146
 Gates, 37
 Generic array logic (GAL), 361
 Glitches, 129

H

Half adders, 374
 Hard array logic (HAL), 362
 Hazards, 129
 Hexadecimal number system, 10
 History of HDLs, 142

I

Idempotent, 87
 Identity theorem, 86
 Input/output blocks (IOBs), 369
 Integrated circuit, 56
 Inverter, 40
 Involution, 88

K

Karnaugh map (K-map), 113

L

Large scale integrated circuit (LSI) logic, 181
 Leading zero, 9
 Least significant bit (LSB), 10
 Light emitting diodes (LEDs), 75
 Logic block (LE), 365
 Logic expression, 38
 Logic families, 56
 Logic function, 38
 Logic HIGH, 45
 Logic levels, 45
 Logic LOW, 45
 Logic minimization, 112
 Logic symbol, 37
 Logic synthesis, 103
 Logic value, 45
 Logic waveform, 39
 Look-up table (LUT), 365

M

Mask read only memory (MROM), 336
 Maxterm list (Π), 108
 Maxterms, 106
 Mealy machine, 224
 Medium scale integrated circuit (MSI) logic, 181
 Memory map model, 331

Metal oxide semiconductor field effect transistor (MOSFET), 56
 Metastability, 200
 Minimal sum, 123, 125
 Minimization, 112
 of logic algebraically, 112
 of logic using K-maps, 116
 Minterm list (Σ), 104
 Minterms, 103
 Modern digital design flow, 149
 Moore machine, 224
 MOSFET. *See* Metal oxide semiconductor field effect transistor (MOSFET)
 Most significant bit (MSB), 10
 Multiplexer design by hand, 190
 Multiplexer modeling in Verilog, 191
 Multiplexers, 190
 Multiplication, 389
 by powers of 2, 392
 combinational multiplier, 391
 shift and add approach, 389
 signed, 393
 unsigned, 389

N

NAND-debounce circuit, 220
 NAND gate, 41
 Negation (\neg), 82
 Negative logic, 45
 Nibble, 10
 NMOS, 57
 Noise, 2
 Noise margin HIGH (MN_H), 47
 Noise margin LOW (MN_L), 47
 Non-volatile memory, 332
 NOR gate, 41
 NPN, 65
 Null element, 86
 Numerals, 7

O

Octal number system, 10
 Ohm's law, 73
 One-hot binary encoder design by hand, 188
 One-hot binary encoder modeling in Verilog, 188
 One-hot decoder design by hand, 182
 One-hot decoder modeling in Verilog, 182
 One's complement numbers, 27
 OR gate, 41
 Output DC specifications. *See* DC specifications
 Output logic macrocell (OLMC), 361

P

7400 Part numbering scheme, 68
 Place and route, 149
 PMOS, 57

PNP, 65
 Positional number system, 7
 Positional weight, 11
 Positive logic, 45
 Postulates, 82
 Power consumption, 4
 Power supplies, 48
 I_{CC} , 48
 I_{GND} , 48
 V_{CC} , 48
 Prime implicant, 117
 Product of sums (POS) form, 94
 Programmable array logic (PAL), 360
 Programmable interconnect points (PIPs), 368
 Programmable logic array (PLA), 359
 Programmable read only memory (PROM), 337
 Proof by exhaustion, 83
 Pull-down network, 58
 Pull-up network, 58

Q

Quiescent current (I_q), 48

R

Radix, 7
 point, 8
 Random access memory (RAM), 333
 Range
 one's complement numbers, 27
 signed magnitude numbers, 25
 two's complement numbers, 29
 Read cycle, 331
 Read only memory (ROM), 332, 333
 Read/write (RW) memory, 332
 Ripple carry adders (RCA), 376
 Ripple counter, 217

S

7-Segment decoder design by hand, 184
 7-Segment decoder modeling in Verilog, 186
 Semiconductor memory, 331
 Sequential access memory, 333
 Sequential logic analysis, 255
 Sequential logic timing, 215
 7400 Series logic families, 67
 Shift register, 222
 Signaling, 1
 Signed magnitude numbers, 25
 Signed numbers, 24
 Simple programmable logic device (SPLD), 363
 Single pole, double throw (SPDT) switch, 218
 Single pole, single throw (SPST) switch, 218
 Sinking current, 46, 47
 Small scale integrated circuit (SSI) logic, 181
 Sourcing and sinking multiple loads, 50
 Sourcing current, 46

Sourcing multiple loads, 50
 SR latch, 202, 205
 SR latch with enable, 208
 Static 0 hazard, 130
 Static 1 hazard, 130
 Static random access memory (SRAM), 342
 Subtraction, 22, 387
 Sum of products (SOP) form, 94
 Switch debouncing, 217
 Switching characteristics, 51
 t_f (fall time), 51
 t_{PHL} (propagation delay HIGH to LOW), 51
 t_{PLH} (propagation delay LOW to HIGH), 51
 t_r (rise time), 51
 t_t (transition time), 51
 Synchronous memory, 335

T

Technology mapping, 149
 Timing hazards, 129
 Toggle flop (T-flop), 216
 Trailing zero, 9
 Transistor-Transistor Logic (TTL), 65
 Transmitter/receiver circuit, 44
 Truth table formation, 38
 TTL operation, 65
 Two's complement arithmetic, 31
 Two's complement numbers, 29

U

Uniting, 93
 Unsigned numbers, 24

V

Verification, 147
 Verilog, 141, 155, 159–163, 170–173, 286, 287, 289,
 303–306, 308, 319, 322, 323, 325, 381
 always blocks, 272
 arrays, 155
 behavioral modeling techniques, 309, 310,
 312, 319–321
 adders, 381
 agents on a bus, 323
 counters, 319
 D-flip-flops, 304
 D-flip-flop with enable, 306
 D-flip-flop with preset, 305
 D-flip-flop with reset, 304
 D-latches, 303
 finite state machines, 308
 encoding styles, 312
 next state logic, 309
 output logic, 310
 state memory, 309
 state variables, 309
 registers, 323
 RTL modeling, 322

- shift registers, 325
 - up counter, 319
 - up counter with range checking, 320
 - up counters with enables, 320
 - up counters with loads, 321
 - case statements, 283
 - casez statements, 283
 - compiler directives, 159
 - include, 159
 - timescale, 159
 - continuous assignment, 164
 - with conditional operators, 165
 - with delay, 167
 - with logical operators, 164
 - counters, 319
 - data types, 153
 - disable, 285
 - drive strength, 153
 - finite state machines, 308
 - for loops, 284
 - forever loops, 283
 - gate level primitives, 172
 - history, 143
 - if-else statements, 280, 281
 - initial blocks, 272
 - net data types, 154
 - number formatting
 - binary, 155
 - decimal, 155
 - hex, 155
 - octal, 155
 - operators, 159
 - assignment, 159
 - bitwise logical, 159
 - bitwise replication, 162
 - Boolean logic, 160
 - concatenation, 161
 - conditional, 161
 - numerical, 162
 - precedence, 163
 - reduction, 160
 - relational, 160
 - parameters, 158
 - procedural assignment, 271
 - procedural blocks, 271
 - repeat loops, 284
 - resolution, 153
 - RTL modeling, 322
 - sensitivity lists, 273
 - signal declaration, 157
 - statement groups, 279
 - structural design and hierarchy, 170
 - explicitly port mapping, 170
 - gate level primitives, 172
 - instantiation, 170
 - positional port mapping, 171
 - user defined primitives, 173
 - system tasks, 286
 - file I/O, 287
 - simulation control, 289
 - text I/O, 286
 - user defined primitives, 173
 - value set, 153
 - variable data types, 154
 - vectors, 154
 - while loops, 283
 - Very large scale integrated circuit (VLSI) logic, 181
 - Volatile memory, 332
- W**
- Weight, 11
 - Word, 10
 - Write cycle, 331
- X**
- X - don't cares, 125
 - XNOR gate, 43
 - XOR gate, 42
 - XOR/XNOR gates in K-maps, 126
- Y**
- Y-chart, 146