

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	17
EXAMPLE 2.7 CONVERTING DECIMAL TO HEXADECIMAL	18
EXAMPLE 2.8 CONVERTING BINARY TO OCTAL	19
EXAMPLE 2.9 CONVERTING BINARY TO HEXADECIMAL	19
EXAMPLE 2.10 CONVERTING OCTAL TO BINARY	20
EXAMPLE 2.11 CONVERTING HEXADECIMAL TO BINARY	20
EXAMPLE 2.12 CONVERTING OCTAL TO HEXADECIMAL	21
EXAMPLE 2.13 CONVERTING HEXADECIMAL TO OCTAL	21
EXAMPLE 2.14 SINGLE-BIT BINARY ADDITION	22
EXAMPLE 2.15 MULTIPLE-BIT BINARY ADDITION	23
EXAMPLE 2.16 SINGLE-BIT BINARY SUBTRACTION	23
EXAMPLE 2.17 MULTIPLE-BIT BINARY SUBTRACTION	24
EXAMPLE 2.18 FINDING THE RANGE OF AN UNSIGNED NUMBER	25
EXAMPLE 2.19 DECIMAL VALUES THAT A 4-BIT, SIGNED MAGNITUDE CODE CAN REPRESENT	26
EXAMPLE 2.20 FINDING THE RANGE OF A SIGNED MAGNITUDE NUMBER	27
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	28
EXAMPLE 2.23 FINDING THE RANGE OF A ONE'S COMPLEMENT NUMBER	29
EXAMPLE 2.24 FINDING THE DECIMAL VALUE OF A ONE'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	31
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	34
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 DE MORGAN'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 DE MORGAN'S THEOREM IN ALGEBRAIC FORM (1)	109
EXAMPLE 4.11 USING DE MORGAN'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

EXAMPLE 4.15 CREATING A CANONICAL SUM OF PRODUCTS LOGIC CIRCUIT USING MINTERMS	117
EXAMPLE 4.16 CREATING A MINTERM LIST FROM A TRUTH TABLE	118
EXAMPLE 4.17 CREATING EQUIVALENT FUNCTIONAL REPRESENTATIONS FROM A MINTERM LIST	119
EXAMPLE 4.18 CREATING A PRODUCT OF SUMS LOGIC CIRCUIT USING MAXTERMS	121
EXAMPLE 4.19 CREATING A MAXTERM LIST FROM A TRUTH TABLE	122
EXAMPLE 4.20 CREATING EQUIVALENT FUNCTIONAL REPRESENTATIONS FROM A MAXTERM LIST	123
EXAMPLE 4.21 CREATING EQUIVALENT FORMS TO REPRESENT LOGIC FUNCTIONALITY	124
EXAMPLE 4.22 MINIMIZING A LOGIC EXPRESSION ALGEBRAICALLY	126
EXAMPLE 4.23 USING A K-MAP TO FIND A MINIMIZED SUM OF PRODUCTS EXPRESSION (2-INPUT)	131
EXAMPLE 4.24 USING A K-MAP TO FIND A MINIMIZED SUM OF PRODUCTS EXPRESSION (3-INPUT)	132
EXAMPLE 4.25 USING A K-MAP TO FIND A MINIMIZED SUM OF PRODUCTS EXPRESSION (4-INPUT)	133
EXAMPLE 4.26 USING A K-MAP TO FIND A MINIMIZED PRODUCT OF SUMS EXPRESSION (2-INPUT)	134
EXAMPLE 4.27 USING A K-MAP TO FIND A MINIMIZED PRODUCT OF SUMS EXPRESSION (3-INPUT)	135
EXAMPLE 4.28 USING A K-MAP TO FIND A MINIMIZED PRODUCT OF SUMS EXPRESSION (4-INPUT)	136
EXAMPLE 4.29 DERIVING THE MINIMAL SUM FROM A K-MAP	138
EXAMPLE 4.30 USING DON'T CARES TO PRODUCE A MINIMAL SOP LOGIC EXPRESSION	139
EXAMPLE 4.31 ELIMINATING A TIMING HAZARD BY INCLUDING NONESSENTIAL PRODUCT TERMS	144
EXAMPLE 5.1 DEFINING VHDL ENTITIES	169
EXAMPLE 5.2 MODELING LOGIC USING CONCURRENT SIGNAL ASSIGNMENTS AND LOGICAL OPERATORS	175
EXAMPLE 5.3 MODELING LOGIC USING CONDITIONAL SIGNAL ASSIGNMENTS	176
EXAMPLE 5.4 MODELING LOGIC USING SELECTED SIGNAL ASSIGNMENTS	178
EXAMPLE 5.5 MODELING LOGIC USING DELAYED SIGNAL ASSIGNMENTS (INERTIAL DELAY MODEL)	179
EXAMPLE 5.6 MODELING LOGIC USING DELAYED SIGNAL ASSIGNMENTS (TRANSPORT DELAY MODEL)	180
EXAMPLE 5.7 MODELING LOGIC USING STRUCTURAL VHDL (EXPLICIT PORT MAPPING)	182
EXAMPLE 5.8 MODELING LOGIC USING STRUCTURAL VHDL (POSITIONAL PORT MAPPING)	183
EXAMPLE 6.1 2-TO-4 ONE-HOT DECODER – LOGIC SYNTHESIS BY HAND	192
EXAMPLE 6.2 3-TO-8 ONE-HOT DECODER – VHDL MODELING USING LOGICAL OPERATORS	193
EXAMPLE 6.3 3-TO-8 ONE-HOT DECODER – VHDL MODELING USING CONDITIONAL AND SELECT SIGNAL ASSIGNMENTS ..	194
EXAMPLE 6.4 7-SEGMENT DISPLAY DECODER – TRUTH TABLE	195
EXAMPLE 6.5 7-SEGMENT DISPLAY DECODER – LOGIC SYNTHESIS BY HAND	196
EXAMPLE 6.6 7-SEGMENT DISPLAY DECODER – MODELING USING LOGICAL OPERATORS	197
EXAMPLE 6.7 7-SEGMENT DISPLAY DECODER – MODELING USING CONDITIONAL AND SELECTED SIGNAL ASSIGNMENTS	198
EXAMPLE 6.8 4-TO-2 BINARY ENCODER – LOGIC SYNTHESIS BY HAND	199
EXAMPLE 6.9 4-TO-2 BINARY ENCODER – VHDL MODELING	200
EXAMPLE 6.10 2-TO-1 MULTIPLEXER – LOGIC SYNTHESIS BY HAND	201
EXAMPLE 6.11 4-TO-1 MULTIPLEXER – VHDL MODELING	202
EXAMPLE 6.12 1-TO-2 DEMULTIPLEXER – LOGIC SYNTHESIS BY HAND	203
EXAMPLE 6.13 1-TO-4 DEMULTIPLEXER – VHDL MODELING	204
EXAMPLE 7.1 PUSH-BUTTON WINDOW CONTROLLER - WORD DESCRIPTION	236
EXAMPLE 7.2 PUSH-BUTTON WINDOW CONTROLLER – STATE DIAGRAM	237
EXAMPLE 7.3 PUSH-BUTTON WINDOW CONTROLLER – STATE TRANSITION TABLE	238
EXAMPLE 7.4 SOLVING FOR THE NUMBER OF BITS NEEDED FOR BINARY STATE ENCODING	240
EXAMPLE 7.5 PUSH-BUTTON WINDOW CONTROLLER – STATE ENCODING	242
EXAMPLE 7.6 PUSH-BUTTON WINDOW CONTROLLER – NEXT STATE LOGIC	243
EXAMPLE 7.7 PUSH-BUTTON WINDOW CONTROLLER – OUTPUT LOGIC	244
EXAMPLE 7.8 PUSH-BUTTON WINDOW CONTROLLER – LOGIC DIAGRAM	245
EXAMPLE 7.9 SERIAL BIT SEQUENCE DETECTOR (PART 1)	246
EXAMPLE 7.10 SERIAL BIT SEQUENCE DETECTOR (PART 2)	247
EXAMPLE 7.11 SERIAL BIT SEQUENCE DETECTOR (PART 3)	248
EXAMPLE 7.12 VENDING MACHINE CONTROLLER (PART 1)	249
EXAMPLE 7.13 VENDING MACHINE CONTROLLER (PART 2)	250
EXAMPLE 7.14 VENDING MACHINE CONTROLLER (PART 3)	251
EXAMPLE 7.15 2-BIT BINARY UP COUNTER (PART 1)	253
EXAMPLE 7.16 2-BIT BINARY UP COUNTER (PART 2)	254
EXAMPLE 7.17 2-BIT BINARY UP/DOWN COUNTER (PART 1)	255
EXAMPLE 7.18 2-BIT BINARY UP/DOWN COUNTER (PART 2)	256
EXAMPLE 7.19 2-BIT GRAY CODE UP COUNTER (PART 1)	257

EXAMPLE 7.20 2-BIT GRAY CODE UP COUNTER (PART 2)	258
EXAMPLE 7.21 2-BIT GRAY CODE UP/DOWN COUNTER (PART 1)	259
EXAMPLE 7.22 2-BIT GRAY CODE UP/DOWN COUNTER (PART 2)	260
EXAMPLE 7.23 3-BIT ONE-HOT UP COUNTER (PART 1)	261
EXAMPLE 7.24 3-BIT ONE-HOT UP COUNTER (PART 2)	262
EXAMPLE 7.25 3-BIT ONE-HOT UP/DOWN COUNTER (PART 1)	263
EXAMPLE 7.26 3-BIT ONE-HOT UP/DOWN COUNTER (PART 2)	264
EXAMPLE 7.27 3-BIT ONE-HOT UP/DOWN COUNTER (PART 3)	265
EXAMPLE 7.28 DETERMINING THE NEXT STATE LOGIC AND OUTPUT LOGIC EXPRESSION OF A FSM	268
EXAMPLE 7.29 DETERMINING THE STATE TRANSITION TABLE OF A FSM	269
EXAMPLE 7.30 DETERMINING THE STATE DIAGRAM OF A FSM	270
EXAMPLE 7.31 DETERMINING THE MAXIMUM CLOCK FREQUENCY OF A FSM	273
EXAMPLE 8.1 BEHAVIOR OF SEQUENTIAL SIGNAL ASSIGNMENTS WITHIN A PROCESS	288
EXAMPLE 8.2 BEHAVIOR OF CONCURRENT SIGNAL ASSIGNMENTS OUTSIDE A PROCESS	288
EXAMPLE 8.3 VARIABLE ASSIGNMENT BEHAVIOR	289
EXAMPLE 8.4 USING IF/THEN STATEMENTS TO MODEL COMBINATIONAL LOGIC	291
EXAMPLE 8.5 USING CASE STATEMENTS TO MODEL COMBINATIONAL LOGIC	293
EXAMPLE 8.6 BEHAVIORAL MODELING OF A RISING EDGE TRIGGERED D-FLIP-FLOP USING ATTRIBUTES	297
EXAMPLE 8.7 CREATING A VHDL TEST BENCH	299
EXAMPLE 8.8 USING REPORT STATEMENTS IN A VHDL TEST BENCH	300
EXAMPLE 8.9 USING ASSERT STATEMENTS IN A VHDL TEST BENCH	301
EXAMPLE 8.10 BEHAVIORAL MODELING OF A D-FLIP-FLOP USING THE RISING_EDGE() FUNCTION	305
EXAMPLE 8.11 WRITING TO AN EXTERNAL FILE FROM A TEST BENCH (PART 1)	314
EXAMPLE 8.12 WRITING TO AN EXTERNAL FILE FROM A TEST BENCH (PART 2)	315
EXAMPLE 8.13 WRITING TO AN EXTERNAL FILE FROM A TEST BENCH (PART 3)	316
EXAMPLE 8.14 WRITING TO STD_OUT FROM A TEST BENCH (PART 1)	317
EXAMPLE 8.15 WRITING TO STD_OUT FROM A TEST BENCH (PART 2)	318
EXAMPLE 8.16 READING FROM AN EXTERNAL FILE IN A TEST BENCH (PART 1)	318
EXAMPLE 8.17 READING FROM AN EXTERNAL FILE IN A TEST BENCH (PART 2)	319
EXAMPLE 8.18 READING FROM AN EXTERNAL FILE IN A TEST BENCH (PART 3)	320
EXAMPLE 8.19 READING SPACE-DELIMITED DATA FROM AN EXTERNAL FILE IN A TEST BENCH (PART 1)	320
EXAMPLE 8.20 READING SPACE-DELIMITED DATA FROM AN EXTERNAL FILE IN A TEST BENCH (PART 2)	321
EXAMPLE 8.21 READING SPACE-DELIMITED DATA FROM AN EXTERNAL FILE IN A TEST BENCH (PART 3)	322
EXAMPLE 9.1 BEHAVIORAL MODEL OF A D-LATCH IN VHDL	329
EXAMPLE 9.2 BEHAVIORAL MODEL OF A D-FLIP-FLOP IN VHDL	330
EXAMPLE 9.3 BEHAVIORAL MODEL OF A D-FLIP-FLOP WITH ASYNCHRONOUS RESET IN VHDL	331
EXAMPLE 9.4 BEHAVIORAL MODEL OF A D-FLIP-FLOP WITH ASYNCHRONOUS RESET AND PRESET IN VHDL	332
EXAMPLE 9.5 BEHAVIORAL MODEL OF A D-FLIP-FLOP WITH SYNCHRONOUS ENABLE IN VHDL	333
EXAMPLE 9.6 PUSH-BUTTON WINDOW CONTROLLER IN VHDL – DESIGN DESCRIPTION	334
EXAMPLE 9.7 PUSH-BUTTON WINDOW CONTROLLER IN VHDL – ENTITY DEFINITION	335
EXAMPLE 9.8 PUSH-BUTTON WINDOW CONTROLLER IN VHDL – ARCHITECTURE	337
EXAMPLE 9.9 PUSH-BUTTON WINDOW CONTROLLER IN VHDL – SIMULATION WAVEFORM	338
EXAMPLE 9.10 PUSH-BUTTON WINDOW CONTROLLER IN VHDL – EXPLICIT STATE CODES	338
EXAMPLE 9.11 SERIAL BIT SEQUENCE DETECTOR IN VHDL – DESIGN DESCRIPTION AND ENTITY DEFINITION	339
EXAMPLE 9.12 SERIAL BIT SEQUENCE DETECTOR IN VHDL – ARCHITECTURE	340
EXAMPLE 9.13 SERIAL BIT SEQUENCE DETECTOR IN VHDL – SIMULATION WAVEFORM	341
EXAMPLE 9.14 VENDING MACHINE CONTROLLER IN VHDL – DESIGN DESCRIPTION AND ENTITY DEFINITION	341
EXAMPLE 9.15 VENDING MACHINE CONTROLLER IN VHDL – ARCHITECTURE	342
EXAMPLE 9.16 VENDING MACHINE CONTROLLER IN VHDL – SIMULATION WAVEFORM	343
EXAMPLE 9.17 2-BIT BINARY UP/DOWN COUNTER IN VHDL – DESIGN DESCRIPTION AND ENTITY DEFINITION	343
EXAMPLE 9.18 2-BIT BINARY UP/DOWN COUNTER IN VHDL – ARCHITECTURE (THREE-PROCESS MODEL)	344
EXAMPLE 9.19 2-BIT BINARY UP/DOWN COUNTER IN VHDL – SIMULATION WAVEFORM	345
EXAMPLE 9.20 4-BIT BINARY UP COUNTER IN VHDL USING THE TYPE UNSIGNED	346
EXAMPLE 9.21 4-BIT BINARY UP COUNTER IN VHDL USING THE TYPE INTEGER	347
EXAMPLE 9.22 4-BIT BINARY UP COUNTER IN VHDL USING THE TYPE STD_LOGIC_VECTOR (1)	348
EXAMPLE 9.23 4-BIT BINARY UP COUNTER IN VHDL USING THE TYPE STD_LOGIC_VECTOR (2)	349
EXAMPLE 9.24 4-BIT BINARY UP COUNTER WITH ENABLE IN VHDL	350

EXAMPLE 9.25 4-BIT BINARY UP COUNTER WITH LOAD IN VHDL	351
EXAMPLE 9.26 RTL MODEL OF AN 8-BIT REGISTER IN VHDL	352
EXAMPLE 9.27 RTL MODEL OF A 4-STAGE, 8-BIT SHIFT REGISTER IN VHDL	353
EXAMPLE 9.28 REGISTERS AS AGENTS ON A DATA BUS – SYSTEM TOPOLOGY	354
EXAMPLE 9.29 REGISTERS AS AGENTS ON A DATA BUS – RTL MODEL IN VHDL	355
EXAMPLE 9.30 REGISTERS AS AGENTS ON A DATA BUS – SIMULATION WAVEFORM	356
EXAMPLE 10.1 CALCULATING THE FINAL DIGIT LINE VOLTAGE IN A DRAM BASED ON CHARGE SHARING	378
EXAMPLE 10.2 BEHAVIORAL MODEL OF A 4x4 ASYNCHRONOUS READ-ONLY MEMORY IN VHDL	383
EXAMPLE 10.3 BEHAVIORAL MODEL OF A 4x4 SYNCHRONOUS READ-ONLY MEMORY IN VHDL	384
EXAMPLE 10.4 BEHAVIORAL MODEL OF A 4x4 ASYNCHRONOUS READ/WRITE MEMORY IN VHDL	386
EXAMPLE 10.5 BEHAVIORAL MODEL OF A 4x4 SYNCHRONOUS READ/WRITE MEMORY IN VHDL	387
EXAMPLE 12.1 DESIGN OF A HALF ADDER	408
EXAMPLE 12.2 DESIGN OF A FULL ADDER	408
EXAMPLE 12.3 DESIGN OF A FULL ADDER OUT OF HALF ADDERS	410
EXAMPLE 12.4 DESIGN OF A 4-BIT RIPPLE CARRY ADDER (RCA)	411
EXAMPLE 12.5 TIMING ANALYSIS OF A 4-BIT RIPPLE CARRY ADDER	412
EXAMPLE 12.6 DESIGN OF A 4-BIT CARRY LOOK AHEAD ADDER (CLA) – OVERVIEW	413
EXAMPLE 12.7 DESIGN OF A 4-BIT CARRY LOOK AHEAD ADDER (CLA) – ALGEBRAIC FORMATION	414
EXAMPLE 12.8 TIMING ANALYSIS OF A 4-BIT CARRY LOOK AHEAD ADDER	415
EXAMPLE 12.9 STRUCTURAL MODEL OF A FULL ADDER IN VHDL USING TWO HALF ADDERS	416
EXAMPLE 12.10 STRUCTURAL MODEL OF A 4-BIT RIPPLE CARRY ADDER IN VHDL	417
EXAMPLE 12.11 VHDL TEST BENCH FOR A 4-BIT RIPPLE CARRY ADDER USING NESTED FOR LOOPS	418
EXAMPLE 12.12 STRUCTURAL MODEL OF A 4-BIT CARRY LOOK AHEAD ADDER IN VHDL	419
EXAMPLE 12.13 4-BIT CARRY LOOK AHEAD ADDER – SIMULATION WAVEFORM	420
EXAMPLE 12.14 BEHAVIORAL MODEL OF A 4-BIT ADDER IN VHDL	421
EXAMPLE 12.15 DESIGN OF A 4-BIT SUBTRACTOR USING FULL ADDERS	422
EXAMPLE 12.16 CREATING A PROGRAMMABLE INVERTER USING AN XOR GATE	423
EXAMPLE 12.17 DESIGN OF A 4-BIT PROGRAMMABLE ADDER/SUBTRACTOR	423
EXAMPLE 12.18 PERFORMING LONG MULTIPLICATION ON DECIMAL NUMBERS	425
EXAMPLE 12.19 PERFORMING LONG MULTIPLICATION ON BINARY NUMBERS	425
EXAMPLE 12.20 DESIGN OF A SINGLE-BIT MULTIPLIER	426
EXAMPLE 12.21 DESIGN OF A 4-BIT UNSIGNED MULTIPLIER	426
EXAMPLE 12.22 TIMING ANALYSIS OF A 4-BIT UNSIGNED MULTIPLIER	427
EXAMPLE 12.23 MULTIPLYING AN UNSIGNED BINARY NUMBER BY TWO USING A LOGICAL SHIFT LEFT	427
EXAMPLE 12.24 ILLUSTRATING HOW AN UNSIGNED MULTIPLIER INCORRECTLY HANDLES SIGNED NUMBERS	428
EXAMPLE 12.25 PROCESS TO CORRECTLY HANDLE SIGNED NUMBERS USING AN UNSIGNED MULTIPLIER	429
EXAMPLE 12.26 PERFORMING LONG DIVISION ON DECIMAL NUMBERS	430
EXAMPLE 12.27 PERFORMING LONG MULTIPLICATION ON BINARY NUMBERS	431
EXAMPLE 12.28 DESIGN OF A 4-BIT UNSIGNED DIVIDER USING A SERIES OF ITERATIVE SUBTRACTORS	432
EXAMPLE 12.29 DIVIDING 1111_2 (15_{10}) BY 0111_2 (7_{10}) USING THE ITERATIVE SUBTRACTION ARCHITECTURE	433
EXAMPLE 12.30 DIVIDING AN UNSIGNED BINARY NUMBERS BY TWO USING A LOGICAL SHIFT RIGHT	434
EXAMPLE 13.1 MEMORY MAP FOR A 256x8 MEMORY SYSTEM	444
EXAMPLE 13.2 EXECUTION OF AN INSTRUCTION TO “LOAD REGISTER A USING IMMEDIATE ADDRESSING”	447
EXAMPLE 13.3 EXECUTION OF AN INSTRUCTION TO “LOAD REGISTER A USING DIRECT ADDRESSING”	448
EXAMPLE 13.4 EXECUTION OF AN INSTRUCTION TO “STORE REGISTER A USING DIRECT ADDRESSING”	449
EXAMPLE 13.5 EXECUTION OF AN INSTRUCTION TO “ADD REGISTERS A AND B”	450
EXAMPLE 13.6 EXECUTION OF AN INSTRUCTION TO “BRANCH ALWAYS”	451
EXAMPLE 13.7 EXECUTION OF AN INSTRUCTION TO “BRANCH IF EQUAL TO ZERO”	452
EXAMPLE 13.8 TOP-LEVEL BLOCK DIAGRAM FOR THE 8-BIT COMPUTER SYSTEM	454
EXAMPLE 13.9 INSTRUCTION SET FOR THE 8-BIT COMPUTER SYSTEM	455
EXAMPLE 13.10 MEMORY SYSTEM BLOCK DIAGRAM FOR THE 8-BIT COMPUTER SYSTEM	456
EXAMPLE 13.11 CPU BLOCK DIAGRAM FOR THE 8-BIT COMPUTER SYSTEM	460
EXAMPLE 13.12 STATE DIAGRAM FOR LDA_IMM	467
EXAMPLE 13.13 SIMULATION WAVEFORM FOR LDA_IMM	468
EXAMPLE 13.14 STATE DIAGRAM FOR LDA_DIR	469
EXAMPLE 13.15 SIMULATION WAVEFORM FOR LDA_DIR	470
EXAMPLE 13.16 STATE DIAGRAM FOR STA_DIR	471

EXAMPLE 13.17 SIMULATION WAVEFORM FOR STA_DIR	472
EXAMPLE 13.18 STATE DIAGRAM FOR ADD_AB	473
EXAMPLE 13.19 SIMULATION WAVEFORM FOR ADD_AB	474
EXAMPLE 13.20 STATE DIAGRAM FOR BRA	475
EXAMPLE 13.21 SIMULATION WAVEFORM FOR BRA	476
EXAMPLE 13.22 STATE DIAGRAM FOR BEQ	477
EXAMPLE 13.23 SIMULATION WAVEFORM FOR BEQ WHEN TAKING THE BRANCH (Z = 1)	478
EXAMPLE 13.24 SIMULATION WAVEFORM FOR BEQ WHEN THE BRANCH IS NOT TAKEN (Z = 0)	479

Appendix B: Concept Check Solutions

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

Index

A

Absorption, 104
Abstraction, 159
AC specifications. *See* Switching characteristics
Adders
 in VHDL, 415
Adder/subtractor circuit, 422
Addition, 22, 407
AND gate, 46
Anti-fuse, 367
Associative property, 102
Asynchronous memory, 365
Axioms, 94
 logical negation, 94
 logical precedence, 95
 logical product, 94
 logical sum, 95
 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, 17
 decimal to octal, 16
 hexadecimal to binary, 20
 hexadecimal to decimal, 14
 hexadecimal to octal, 21
 octal to binary, 20
 octal to decimal, 13
 octal to hexadecimal, 20
Binary addition. *See* Addition
Binary number system, 9
Binary subtraction. *See* Subtraction
Bipolar junction transistor (BJT), 71
Bistable, 212
Boolean algebra, 93
Boolean algebra theorems, 95
Borrows, 23
Break-before-make switch behavior, 232
Buffer, 45
Byte, 10

C

Canonical product of sums, 119
Canonical sum of products, 115

Capacity, 361
Carry, 22
Carry look ahead adders (CLA), 412
Charge sharing, 377
Classical digital design flow, 162
CMOS. *See* Complementary metal oxide semiconductors (CMOS)
Combinational logic analysis, 111
Combining, 105
Commutative property, 100
Complementary metal oxide semiconductors (CMOS), 4, 62
 gates, 64
 inverter, 64
 NAND gate, 65
 NOR gate, 68
 operation, 63
Complements, 99
Complete sum, 138
Complex programmable logic device (CPLD), 397
Computer system design, 439
 addressing modes, 445
 arithmetic logic unit (ALU), 441
 central processing unit, 441
 condition code register, 441
 control unit, 441
 data memory, 440
 data path, 441
 direct addressing, 446
 example 8-bit system, 453
 control unit, 464
 CPU, 460
 data path, 461
 detailed instruction execution, 466
 instruction set, 454
 memory system, 455
 general purpose registers, 441
 hardware, 439
 immediate addressing, 446
 indexed addressing, 446
 inherent addressing, 446
 input output ports, 440
 instructions, 439
 branches, 450
 data manipulations, 449
 loads and stores, 447
 register, 441
 memory address register, 441
 memory map, 444
 memory mapped system, 442
 opcodes, 445
 operands, 445

Computer system design (*cont.*)
 program, 439

- program counter, 441
- program memory, 440
- registers, 441
- software, 439, 445

Configurable logic block (CLB), 399

Conjunction (\wedge), 94

Converting between bases. *See* Base conversions

Converting between positive and negative logic, 96

Counters, 253, 345

- designing by hand, 253
- modeling in VHDL, 345

Covering, 104

Cross-coupled inverter pair, 211

D

Data sheet, 57

DC specifications, 52

- I_{IH-max} , 53
- I_{IL-max} , 53
- I_{I-max} , 53
- I_{OH-max} , 52
- I_{OL-max} , 52
- I_{O-max} , 52
- I_q (quiescent current), 54
- NM_H , 53
- NM_L , 53
- V_{IH-max} , 53
- V_{IH-min} , 53
- V_{IL-max} , 53
- V_{IL-min} , 53
- V_{OH-max} , 52
- V_{OH-min} , 52
- V_{OL-max} , 52
- V_{OL-min} , 52

Decimal number system, 9

Decoders, 191

De Morgan's Theorem of Duality, 95

De Morgan's Theorems, 105

Demultiplexer design by hand, 203

Demultiplexer modeling in VHDL, 204

Demultiplexers, 203

Design abstraction, 159

Design domains, 160

- behavioral, 160
- physical, 160
- structural, 160

Design levels, 160

- algorithmic, 160
- circuit, 160
- gate, 160
- register transfer, 160
- system, 160

Design simplicity, 4

D-Flip-Flops, 223

Digit, 9

Digital design flow, 162

Digit notation, 9

Diodes, 81

Discrete components, 62

Disjunction (\vee), 94

Distinguished one cells, 138

Distributive property, 103

Division, 430

- by powers of 2, 434
- signed, 434
- unsigned, 430
- using iterative subtractions, 431

D latch, 222

Don't cares (X), 139

Double pole, double throw (DPDT) switch, 231

Double pole, single throw (DPST) switch, 231

Driving loads, 77

- driving LEDs, 82
- driving resistive loads, 79

Dual in-line package (DIP), 75

Duality, 95

Dynamic hazard, 143

Dynamic random access memory (DRAM), 375

E

Electrically erasable programmable read-only memory (EEPROM), 370

Electrical signaling, 1

Encoders, 199

Erasable programmable read-only memory (EPROM), 368

Essential prime implicant, 138

F

Field programmable gate array (FPGA), 398

Finite state machines (FSM), 236

- behavioral modeling in VHDL, 334
- binary state encoding, 239
- design examples by hand, 246
- design process, 245
- final logic diagram, 244
- gray code state encoding, 240
- introduction, 236
- next state logic, 243
- one-hot state encoding, 241
- output logic, 243
- reset condition, 266
- state diagram, 236
- state memory, 239
- state transition table, 238
- state variables, 242
- synthesis by hand, 238

FLASH memory, 371

- NAND-FLASH, 371
- NOR-FLASH, 371

Floating-gate transistor, 368
 Forward current (I_F), 81
 Forward voltage (V_F), 81
 Full adders, 408
 Functionally complete sets, 110
 Fuse, 367

G

Gajski and Kuhn's Y-chart, 160
 Gates, 43
 Generic array logic (GAL), 395
 Glitches, 142

H

Half adders, 408
 Hard array logic (HAL), 396
 Hazards, 142
 Hexadecimal number system, 10
 History of HDLs, 156

I

Idempotent, 99
 Identity theorem, 98
 Input/output blocks (IOBs), 403
 Integrated circuit, 62
 Inverter, 46
 Involution, 100

K

Karnaugh map (K-map), 126

L

Large scale integrated circuit (LSI) logic, 191
 Leading zero, 9
 Least significant bit (LSB), 10
 Light emitting diodes (LEDs), 81
 Logic block (LE), 399
 Logic expression, 44
 Logic families, 62
 Logic function, 44
 Logic HIGH, 51
 Logic levels, 51
 Logic LOW, 51
 Logic minimization, 125
 Logic symbol, 43
 Logic synthesis, 115
 Logic value, 51
 Logic waveform, 45
 Look-up table (LUT), 399

M

Mask read-only memory (MROM), 366
 Maxterm list (M), 122
 Maxterms, 119

Mealy machine, 237
 Medium scale integrated circuit (MSI) logic, 191
 Memory map model, 361
 Metal oxide semiconductor field effect transistor (MOSFET), 62
 Metastability, 212
 Minimal sum, 137, 138
 Minimization, 125
 Minimization of logic algebraically, 125
 Minimization of logic using K-maps, 129
 Minterm list (Σ), 118
 Minterms, 115
 Modern digital design flow, 162
 Moore machine, 237
 MOSFET. *See* Metal oxide semiconductor field effect transistor (MOSFET)
 Most significant bit (MSB), 10
 Multiplexer design by hand, 201
 Multiplexer modeling in VHDL, 202
 Multiplexers, 201
 Multiplication, 424

- combinational multiplier, 426
- by powers of 2, 427
- shift and add approach, 424
- signed, 428
- unsigned, 424

N

NAND-debounce circuit, 233
 NAND gate, 47
 Negation (\neg), 94
 Negative logic, 51
 Nibble, 10
 NMOS, 63
 Noise, 3
 Noise margin HIGH (MN_H), 53
 Noise margin LOW (MN_L), 53
 Non-volatile memory, 362
 NOR gate, 47
 NPN, 71
 Null element, 98
 Numerals, 7

O

Octal number system, 10
 Ohm's law, 79
 One-hot binary encoder design

- by hand, 199

 One-hot binary encoder modeling in VHDL, 199
 One-hot decoder design by hand, 191
 One-hot decoder modeling in VHDL, 192
 One's complement numbers, 28
 OR gate, 47
 Output DC specifications. *See* DC specifications
 Output logic macrocell (OLMC), 395

P

Place and route, 163
 PMOS, 63
 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, 130
 Product of sums (POS) form, 106
 Programmable array logic (PAL), 394
 Programmable interconnect points (PIPs), 402
 Programmable logic array (PLA), 393
 Programmable read-only memory (PROM), 367
 Proof by exhaustion, 95
 Pull-down network, 64
 Pull-up network, 64

Q

Quiescent current (I_q), 54

R

Radix, 7
 Radix point, 8
 Random access memory (RAM), 362
 Range
 one's complement numbers, 29
 signed magnitude numbers, 27
 two's complement numbers, 31
 unsigned numbers, 25, 27, 29, 31
 Read cycle, 361
 Read-only memory (ROM), 362, 363
 Read/write (RW) memory, 362
 Ripple carry adders (RCA), 410
 Ripple counter, 229

S

Semiconductor memory, 361
 Sequential access memory, 362
 Sequential logic analysis, 267
 Sequential logic timing, 227
 7-Segment decoder design by hand, 195
 7-Segment decoder modeling in VHDL, 196
 7400 DC operating conditions, 74
 7400 DIP pin-out, 75
 7400 Part numbering scheme, 73
 7400 Series logic families, 73
 Shift register, 235
 Signaling, 1
 Signed magnitude numbers, 26

Signed numbers, 26
 Simple programmable logic device (SPLD), 397
 Single pole, double throw (SPDT) switch, 231
 Single pole, single throw (SPST) switch, 231
 Sinking current, 52, 53
 Small scale integrated circuit (SSI) logic, 191
 Sourcing and sinking multiple loads, 56
 Sourcing current, 52
 Sourcing multiple loads, 55
 SR latch, 214, 217
 SR latch with enable, 220
 Static 0 hazard, 143
 Static 1 hazard, 143
 Static random access memory (SRAM), 372
 Subtraction, 23, 422
 Sum of products (SOP) form, 106
 Switch debouncing, 230
 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, 365

T

Technology mapping, 163
 Timing hazards, 142
 Toggle flop (T-flop), 228
 Trailing zero, 9
 Transistor-transistor logic (TTL), 71
 operation, 71
 Transmitter/receiver circuit, 50
 Truth table formation, 44
 Two's complement arithmetic, 33
 Two's complement numbers, 30

U

Uniting, 105
 Unsigned numbers, 25

V

Verification, 161
 Verilog HDL, 157
 Very large scale integrated circuit (VLSI) logic, 191
 VHDL, 155
 VHDL behavioral modeling techniques
 adders, 415
 counters, 345
 with enables, 349
 with loads, 350
 using type INTEGER, 347
 using type STD_LOGIC_VECTOR, 347
 using type UNSIGNED, 346
 D-Flip-Flops, 330
 D-latches, 329

- finite state machines, 334
 - design examples, 339
 - explicit state encoding using subtypes, 338
 - three process model, 335
 - user-enumerated state encoding, 335
- modeling agents on a bus, 354
- modeling memory, 382
- modeling registers, 352
- modeling shift registers, 353
- RTL modeling, 352
- VHDL constructs, 165
 - architecture, 165, 169
 - assignment operator (\leftarrow), 171
 - attributes, 296
 - case statements, 292
 - component declaration, 170
 - concatenation operator, 174
 - concurrent signal assignments, 174
 - concurrent signal assignments with logical operators, 174
 - conditional signal assignments, 175
 - constant declaration, 170
 - data types, 166
 - delayed signal assignments, 178
 - inertial, 179
 - transport, 179
 - entity, 165
 - entity definition, 168
 - for loops, 295
 - if/then statements, 290
 - libraries and packages, 168
 - logical operators, 172
 - loop statements, 294
 - numerical operators, 173
 - operators, 171
 - packages, 165
 - process, 285
 - sensitivity list, 285
 - wait statement, 286
 - relational operators, 173
 - selected signal assignments, 177
 - sequential signal assignments, 287
 - shift operators, 173
 - signal declaration, 169
 - structural design, 181
 - component declaration, 170
 - component instantiation, 181
 - explicit port mapping, 181
 - port mapping, 181
 - positional port mapping, 182
- test benches, 298
 - assert statements, 300
 - reading/writing external files, 311
 - report statements, 299

- variables, 289
- while loops, 295
- VHDL data types
 - array, 167
 - bit, 166
 - bit_vector, 167
 - Boolean, 166
 - character, 166
 - integer, 166
 - natural, 168
 - positive, 168
 - real, 166
 - std_logic, 302
 - std_logic_vector, 302
 - std_ulogic, 302
 - std_ulogic_vector, 302
 - string, 167
 - time, 167
 - user-defined enumerated, 167
- VHDL packages, 302
 - MATH_COMPLEX, 311
 - MATH_REAL, 309
 - NUMERIC_BIT, 308
 - NUMERIC_BIT_UNSIGNED, 309
 - NUMERIC_STD, 306
 - conversion functions, 307
 - type casting, 307
 - NUMERIC_STD_UNSIGNED, 308
 - standard, 165
 - STD_LOGIC_1164, 302
 - resolution function, 303
 - type conversions, 305
 - STD_LOGIC_ARITH, 322
 - STD_LOGIC_SIGNED, 323
 - STD_LOGIC_TEXTIO, 311
 - STD_LOGIC_UNSIGNED, 323
 - TEXTIO, 311
- Volatile memory, 362

W

- Weight, 11
- Word, 10
- Write cycle, 361

X

- X-don't cares, 139
- XNOR gate, 49
- XOR gate, 48
- XOR/XNOR gates in K-maps, 140

Y

- Y-chart, 160