Index

A

Abstraction, 15–16
  behavioral (algorithmic) level, 15–16
  dataflow level, 16
  gate level, 16
  switch level, 16
acc_close(), 284–85
Access routines, 273, 283–91, 369–77
  examples of, 285–91
  fetch routines, 373–75
  handle routines, 369–71
  mechanics of, 284–85
  modify routines, 377
  next routines, 371–73
  types of, 285
  utility access routines, 376–77
  Value Change Link (VCL) routines, 373
acc_initialize(), 284–85
Advanced net types, 364–67
  supply1 and **supply0** keywords, 366
  trand keyword, 366–67
  tri keyword, 364–65
  tri0 keyword, 366
  tri1 keyword, 366
  trior keyword, 366–67
  trireg keyword, 365
  wand keyword, 366–67
  wor keyword, 366–67
Advanced verification techniques, 341–60
  assertion checking, 352–54
    assertion-based verification (ABV), advantages of, 354
    static assertions, 353
    temporal assertions, 353
  formal verification, 355–58
    equivalence checking, 358
    semi-formal verification, 356–57
  traditional verification flow, 342–52
    analysis, 350–51
    architectural modeling, 343–44
    block level verification, 345
    coverage, 351–52
    design-under-test (DUT), 343
    extended verification, 345
    full chip verification, 345
    functional verification environment, 344–47
    high-level verification languages (HVLs), 346–47
    simulation, 347–50
Algorithmic level, abstraction, 15
always statement, 123–24, 310–11
and/or gates, 64–66
Architectural modeling, 343–44
Area, as design constraint, 314
Arithmetic operators, 97–98, 306
  binary operators, 97–98
  unary operators, 98
Arrays, 37–38
Assertion-based verification (ABV), advantages, 354
Assertion checking, 8, 352–54
  static assertions, 353
  temporal assertions, 353
Assertion checking tools, 424
assign keyword, 187–89
assign statements, 324–25
Attributes, 323*fn*, 417
Automated logic synthesis, 302–4
Automatic (re-entrant) tasks, 176–77
Automatic (recursive) functions, 181–82

B

Backus-Naur Form (BNF), 389–419
Behavioral (algorithmic) level, abstraction, 15
Behavioral DRAM model, example, 434–38
Behavioral modeling, 119–70
  blocks, 146–51
    named, 150–51
    nested, 149–50
    parallel, 146, 148–49
    sequential, 146–48
    special features of, 149–51
  conditional statements, 137–38
  examples, 159–66
    4-bit counter, 160
    4-to-1 multiplexer, 159–60
    traffic signal controller, 161–66
  loops, 141–46
    forever loop, 146
    for loop, 143–44
    repeat loop, 144–45
    while loop, 141–43
  multiway branching, 138–41
    case statement, 138–40
    casex keyword , 141
    casez keyword , 141

procedural assignments, 124–29
blocking assignments, 125–26
nonblocking assignments, 126–29
structured procedures, 120–24
always statement, 123–24
initial statement, 120–23
timing controls, 129–36
delay-based timing control, 129–32
event-based timing control, 133–36
level-sensitive timing control, 136
Behavioral statements, 402–6
case statements, 405–6
conditional statements, 405
continuous assignment statements, 402
looping statements, 406
parallel blocks, 403
procedural blocks and assignments, 403
sequential blocks, 403
statements, 403–4
task enable statements, 406
timing control statements, 404–5
Behavioral synthesis tools, 6
Bidirectional switches, 238–39
Binary operators, 28, 97–98
Bit-wise operators, 306
Bitwise operators, 101–2
Blank spaces, 27
Block item declarations, 398
Block level verification, 345
Blocking assignments, 125–26
Blocks, 146–51
generate statements, 152–58
generate case, 158
generate conditional, 156–57
generate loop, 153–56
named, 150–51
nested, 149–50
parallel, 146, 148–49
sequential, 146–48
special features of, 149–51
Bottom-up design methodology, 11–13
Bridging faults, 422
buf/notgates, 66–69
butif/notif gates, 68–69
gate instantiations for, 69
truth tables for, 69

C

Carry lookahead mechanism, 108
case statement, 138–40, 405–6
casex keyword , 141
casez keyword , 141
Cell characterization, 313
clk signal, 18–19, 21
Closing files, 201
cmos keyword, 237–38, 241
CMOS switches, 237–38
delay specification on, 241
Combinational UDPs, 251
definition, 253–55
example of, 258–60
instantiating UDP primitives, 257
shorthand notation for don't cares, 256
state table entries, 255–56
Comments, 28, 417
Compiled code simulators, 422
Compiler directives, 44–45, 387
Computer-aided digital design:
evolution of, 3–4
HDLs:
advantages of, 7
emergence of, 4
importance of, 7
trends in, 8–9
typical design flow, 5–6
Verilog HDL, 4
popularity of, 7–8
Computer-Aided Engineering (CAE) tools, 3*fn*
Computer-aided logic synthesis tools, 301
Concatenation operator, 103
Concatenations, 412
Conditional operator, 104–5
Conditional statements, 137–38, 405
Configuration source text, 390
Constant functions, 182–83
Continuous assignment statements, 402
Continuous assignments, 90–92
Coverage, 351–52
functional coverage, 352
structural coverage, 351–52
Coverage tools, 424
Cycle simulation, 422

D

D-flipflops (D_FFs), 13–14, 17, 21
Data checkers, 351
Data types, 32–40
arrays, 37–38
declarations, 394–95
integer, 36
memories, 38
nets, 33
parameters, 39
real, 36
registers, 33–34
strings, 40
time, 37
value set, 32–33
vectors, 34–35
Dataflow level, abstraction, 16
Dataflow modeling, 89–118
continuous assignments, 90–92
delays, 92–94
examples, 106–15
4-bit full adder, 108–10
4-to-1 multiplexer, 106–8
ripple counter, 110–15
expressions, 94
logic synthesis, 89
operands, 95
operator types, 96–106
operators, 95
d_clk signal, 19
deassign keyword, 187–89
Decimal notation, 36
Declarations, 393–98
assignments, 396
block item declarations, 398
data types, 394–95
declaration types, 393–94
delays, 395

function declarations, 396–97
lists, 395–96
module parameter declarations, 393
net and variable types, 394–95
port declarations, 393
ranges, 396
strengths, 395
task declarations, 397
type declarations, 394
'**define** directive, 44
defparam statement, 190–92
Delay back-annotation, 230–32
Delay-based timing control:
intra-assignment delay control, 131–32
regular delay control, 130–31
zero delay control, 132
Delay models, 216–19
distributed delays, 216–17
lumped delays, 217–18
pin-to-pin delays, 218–19
Delays, 92–94
Design block, example, 20–21
Design constraint specification, 329
Design flow, 5–6
Design methodologies, 11–13
Design partitioning, 327–29
horizontal partitioning, 327–28
parallelizing design structure, 328–29
vertical partitioning, 328
Design reuse, 304
Design-under-test (DUT), 343
disable keyword, 151
$display task, 201–2
Distributed delays, 216–17
d_reset signal, 19
$dumpall task, 206–7
$dumpfile task, 206–7
$dumpoff task, 206–7
$dumpon task, 206–7
$dumpvars task, 206–7

E

EDA tools, 3*fn*, 6
Edge-sensitive sequential UDPs, 263–64
Electronic Design Automation (EDA), 3
Elements, 38
else keyword, 137
endtask keyword, 172
Equality operators, 100, 306
Equivalence checking, 322, 358
Equivalence checking tools, 424
Escaped identifiers, 31
Event-based timing control:
event OR control, 134–36
named event control, 133–34
regular event control, 133
Event-driven simulation, 422
Event OR control, 134–36
Expressions, 94, 412–17
concatenations, 412
function calls, 412–13
left-side values, 415
numbers, 416–17
operators, 415–16
primaries, 414–15
strings, 417
Extended verification, 345

F

Fall delay, 225–26
gates, 79
Fault simulation, 422
$fdisplay task, 200–201
Fetch routines, 285, 373–75
$fflush task, 200*fn*
$fgetc task, 200*fn*
File output:
closing files, 201
opening a file, 199–200
writing to files, 200–201
$finish task, 292
Finite state machine (FSM), 330–31
$fmonitor task, 200–201
$fopen task, 199
for loop, 143–44
force keyword, 189–90
on nets, 190
forever loop, 146
Formal syntax definition, 389–419
behavioral statements, 402–6
case statements, 405–6
conditional statements, 405
continuous assignment statements, 402
looping statements, 406
parallel blocks, 403
procedural blocks and assignments, 403
sequential blocks, 403
statements, 403–4
task enable statements, 406
timing control statements, 404–5
declarations, 393–98
assignments, 396
block item declarations, 398
data types, 394–95
declaration types, 393–94
delays, 395
function declarations, 396–97
lists, 395–96
module parameter declarations, 393
net and variable types, 394–95
port declarations, 393
ranges, 396
strengths, 395
task declarations, 397
type declarations, 394
expressions, 412–17
concatenations, 412
function calls, 412–13
left-side values, 415
numbers, 416–17
operators, 415–16
primaries, 414–15
strings, 417
general, 417–19
attributes, 417
comments, 417
identifier branches, 419
identifiers, 417–19
whitespace, 419
generated instantiation, 400–401
module instantiation, 400

primitive instances, 398–400
gate and switch types, 400
instantiation and instances, 398–99
strengths, 399
terminals, 399
source text, 390–92
configuration source text, 390
library source text, 390
module and primitive source text, 390–91
module items, 391–92
module parameters and ports, 391
specify section, 406–12
specify block declaration, 406–7
specify block terminals, 407–8
specify path declaration, 407
specify path delays, 408–9
system timing checks, 409–12
UDP body, 402
UDP declaration, 401
UDP instantiation, 402
UDP ports, 401–2
Formal verification, 355–58
equivalence checking, 358
semi-formal verification, 356–57
Formal verification techniques, 8
Formal verification tools, 424
FPGAs (Field Programmable Gate Arrays), 4
$fread task, 200*fn*
$fscanf task, 200*fn*
$fseek task, 200*fn*
$fstrobe task, 200–201
$ftell task, 200*fn*
Full chip verification, 345
Function calls, 412–13
Function declarations, 396–97
function statement, 311
Functional coverage, 352
Functional verification environment, 344–47
Functions:
automatic (recursive), 181–82
constant, 182–83
declaration/invocation, 178
endfunction keyword, 177
examples, 179–81
left/right shifter, 180–81
parity calculation, 179–80
function keyword, 177
signed, 183
syntax for, 178
tasks compared to, 171–72
when to use, 177–78
$fwrite task, 200–201

G

Gate delays, 79–84
example, 82–84
fall delay, 79
max value, 80–81
min value, 80–81
rise delay, 79
turn-off delay, 79
typ value, 80–81
Gate level, abstraction, 16
Gate-level modeling, 63–87
gate delays, 79–84
gate types, 64–78
Gate-level netlist, 9
Gate-level netlist verification, 320–23
functional verification, 320–22
timing verification, 323
Gate types, 64–78
and/or gates, 64–66
array of instances, 70
buf/notgates, 66–69
examples, 71–78
4-bit ripple carry full adder, 75–78
gate-level multiplexer, 71–74
Gateway Design Automation, 4
Generate statements, 152–58
generate case, 158
generate conditional, 156–57
generate loop, 153–56
"GIGO "Garbage In Garbage Out" phenomenon, 6

H

handle keyword, 285
Handle routines, 285, 369–71
Hardware acceleration, 348–49
Hardware acceleration tools, 423
Hardware Description Languages (HDLs), *See* HDLs
Hardware emulation, 349–50
HDLs:
emergence of, 4
trends in, 8–9
Hierarchical modeling, 11–25
design methodologies, 11–13
example, 20–23
design block, 20–21
stimulus block, 20–23
4-bit ripple carry counter, 13–14
instances, 15–17
modules, 14–16
simulation, components of, 18–19
Hierarchical name referencing, 59
Hierarchical names, 59–60
High-level description, 300
High-level design, 303–4
High-level verification languages (HVLs), 346–47, 423
$hold task, 229
Horizontal partitioning, 327–28

I

Identifier branches, 419
Identifiers, 31, 59, 417–19
IEEE Standard Verilog Hardware Description Language, 200*fn*, 230, 235, 323*fn*
if keyword, 137
'ifdef directive, 194–95
'ifndef directive, 194–95
In-circuit emulation tools, 424
'**include** directive, 44
init.dat, 205
initial statement, 120–23
combined ANSI C style port declaration and initialization, 122–23

combined port/data declaration and initialization, 122
combined variable declaration and initialization, 121–22
Instances, 15–17
Instantiation, 15–16
integer data type, 36
Integrated circuits, 3–4
Interpreted simulators, 422
Inverters, 13
Invoking PLI tasks, 278

K

Keywords, 31, 385–86

L

Left-side values, 415
Level-sensitive sequential UDPs, 261–62
Lexical conventions, 27–47
comments, 28
escaped identifiers, 31
identifiers, 31
keywords, 31
number specification, 28–30
operators, 28
strings, 30
whitespace, 27
Library cells, 313
Library source text, 390
Linking PLI tasks, 277
Local parameters, 39
localparam keyword, 39
Log files, 351
Logic optimization, 313
Logic synthesis, 89, 299–340
automated, 302–4
computer-aided logic synthesis tools, 301
defined, 300–303
gate-level netlist verification, 320–23
functional verification, 320–22
timing verification, 323
impact of, 303–4
modeling tips for, 323–29
design constraint specification, 329
design partitioning, 327–29
Verilog coding style, 323–27
sequential circuit synthesis example, 330–38
circuit requirements, 330
design constraints, 334
design specification, 330
finite state machine (FSM), 330–31
logic synthesis, 334
optimized gate-level netlist, 335–36
technology library, 334
verification, 337–38
Verilog description, 331–34
synthesis design flow, 311–20
RTL to gates, 311–20
Verilog HDL synthesis, 304–11
Verilog constructs, 305–11
Verilog operators, 306–7
Logic synthesis tools, 6, 304
support for Verilog HDL, 8
Logical operators, 98–99, 306
Looping statements, 406
Loops, 141–46
forever loop, 146
for loop, 143–44
repeat loop, 144–45
while loop, 141–43
LSI (Large Scale Integration) chips, 3
Lumped delays, 217–18

M

Max delays, 225–26
max value, 80–81
+maxdelays option, 227
Memories, 38
Min delays, 225–26
min value, 80–81
+mindelays option, 227
Mixed bottom-up methodology, 9
Modeling techniques, 187–212
assign keyword, 187–89
conditional compilation, 194–95
conditional execution, 195–96
deassign keyword, 187–89
force keyword, 189–90
overriding parameters, 190–93
defparam statement, 190–92
module-instance parameter values, 192–93
procedural continuous assignments, 187–90
release keyword, 189–90
system tasks, 199–208
displaying hierarchy, 201–2
file output, 199–201
initializing memory from file, 204–6
random number generation, 203–4
strobing, 202–3
value change dump (VCD) file, 206–8
time scales, 197–98
Modeling tips for logic synthesis, 323–29
design constraint specification, 329
design partitioning, 327–29
Verilog coding style, 323–27
Modify routines, 285, 377
Module and primitive source text, 390–91
Module-instance parameter values, 192–93
Module items, 391–92
Module parameters, 391
Module_name, 15
Modules, 14–16, 49–52
behavioral blocks, 50
components of, 50
conceptual internal representation of, 280
dataflow statements, 50
definitions, 17–18, 50
instantiation of lower modules, 50
module name, 50

port declarations, 50, 54–55
port list, 50, 53
tasks/functions, 50
variable declarations, 50
Module_terminal_list, 15
$monitor task, 201–2
MOS switches, 236–37
delay specification on, 241
MSI {Medium Scale Integration) chips, 3
Multiline comments, 28
Multiway branching, 138–41
case statement, 138–40
casex keyword , 141
casez keyword , 141

N

Named blocks, 150–51
disabling, 151
Named event control, 133–34
Native compiled code simulators, 422
Negative edge-triggered toggle flipflops (T_FF), 13
Negative numbers, 30
Nest modules, 17–18
Nested blocks, 149–50
Nets, 33
Newlines, 27
Next routines, 285, 371–73
nmos keyword, 236, 241
Nonblocking assignments, 126–29
application of, 127–29
Number specification, 28–30
negative numbers, 30
question marks, 30
sized numbers, 28–29
underscore characters, 30
unsized numbers, 29
X or Z values, 29
Numbers, 416–17

O

Object types, 279–82
One-line comments, 28
Opening a file, 199–200
Operands, 95
Operating environment factors, 314
Operator precedence, 105–6
Operator types, 96–106
arithmetic operators, 97–98
binary operators, 97–98
unary operators, 98
bitwise operators, 101–2
concatenation operator, 103
conditional operator, 104–5
equality operators, 100
logical operators, 98–99
operator precedence, 105–6
reduction operators, 102
relational operators, 99–100
replication operator, 104
shift operators, 103
symbols, 96–97
Operators, 28, 95, 415–16
Optimized internal representation, 313
Overriding parameters, 190–93
defparam statement, 190–92
module-instance parameter values, 192–93

P

PALS (Programmable Array Logic), 4
Parallel blocks, 146, 148–49, 403
Parallelizing design structure, 328–29
parameter keyword, 39, 224
Parameters, 39
Path delay modeling, 219–27
conditional path delays, 224–25
edge-sensitive paths, 223
fall delays, 225–26
full connection, 222–23
handling **x** transitions, 227
max delays, 225–26
min delays, 225–26
parallel connection, 221
rise delays, 225–26
specify blocks, 219–27
specparam statement, 223–24
turn-off delays, 225–26
typical delays, 226–27
Pin-to-pin delays, 218–19
pmos keyword, 236, 241
Port connection rules, 56–57
illegal port connection, example of, 57
inouts, 56
inputs, 56
outputs, 56
unconnected ports, 57
width matching, 56
Port declarations, 393
Ports, 52–60, 391
connecting to external signals, 57–59
connecting by name, 58–59
connecting by ordered list, 58
connection rules, 56–57
declarations, 50, 54–55
list of, 50, 53
Postprocessing tools, 206
Power, as design constraint, 314
Primaries, 414–15
Primitive instances, 398–400
gate and switch types, 400
instantiation and instances, 398–99
strengths, 399
terminals, 399
Procedural assignments, 124–29
blocking assignments, 125–26
nonblocking assignments, 126–29
Procedural blocks and assignments, 403
Programming Language Interface (PLI), 8, 273–97
access routines, 369–77
fetch routines, 373–75
handle routines, 369–71
modify routines, 377
next routines, 371–73
utility access routines, 376–77
Value Change Link (VCL) routines, 373
internal data representation, 279–83
invoking PLI tasks, 278
library routines, 283–95
access routines, 283–91
utility routines, 291–95

linking PLI tasks, 277
routines, 369–84
conventions, 369
task addition/invocation, general flow of, 279
uses of, 276
utility routines, 378–83
Display Messages, 382
Get Argument List Information, 378
Get Calling Task/Function Information, 378
Get Parameter Values, 379
housekeeping tasks, 383
Long Arithmetic, 382
miscellaneous, 383
Monitor Parameter Value Changes, 380
Put Parameter Value, 379
Synchronize Tasks, 380–81
Programming Language Interface (PLI) Manual, 284
Protocol checkers, 351

Q

Question marks, 30

R

rcmos keyword, 241
Re-entrant tasks, 176–77
real data type, 36
Recursive functions, 181–82
Reduction operators, 102
Register transfer level (RTL), 4, 15, 304–5
Registers, 33–34
Regular event control, 133
Relational operators, 99–100, 306
release keyword, 189–90
on nets, 190
repeat loop, 144–45
Replication operator, 104
reset signal, 18–19, 21
Reuse, 304
$rewind task, 200*fn*
Ripple carry counter, 15, 20
Rise delay, 225–26
gates, 79
rnmos keyword, 241
Root module, 59–60
rpmos keyword, 241
RTL level, designing in HDL at, 8
RTL to gates, 311–20
design constraints, 314
example of, 315–20
logic optimization, 313
optimized gate-level description, 315
RTL description, 312
technology library, 313–14
technology mapping and optimization, 313
translation, 312–13
unoptimized intermediate representation, 313

S

Scientific notation, 36
Scoreboards, 351
Semi-formal verification, 356–57
Sequential blocks, 146–48, 403
Sequential circuit synthesis example, 330–38
circuit requirements, 330
design constraints, 334
design specification, 330
finite state machine (FSM), 330–31
logic synthesis, 334
optimized gate-level netlist, 335–36
technology library, 334
verification, 337–38
Verilog description, 331–34
Sequential UDPs, 251
compared to combinational UDPs, 260
edge-sensitive, 263–64
example of, 265–66
level-sensitive, 261–62
$setup task, 228–29
Shift operators, 103
Signal contention, 364
Signed functions, 183
Signed quantities, 36
$signed task, 199
Signed variables, 34
Simulation, 347–50
hardware acceleration, 348–49
hardware emulation, 349–50
software simulation, 347
Simulation seconds, 37
Simulation time, 37
Simulation tools, 423
Sized numbers, 28–29
Software simulation, 347
Source text, 390–92
configuration source text, 390
library source text, 390
module and primitive source text, 390–91
module items, 391–92
module parameters and ports, 391
Special characters, in strings, 40
Special features of blocks, 149–51
Specify blocks, 219–27
delay specification on, 242
edge-sensitive paths, 223
full connection, 222–23
parallel connection, 221
specparam keyword, 223
specparam statement, 223–24
SR latch, 50–52
$sscanf task, 200*fn*
SSI (Small Scale Integration) chips, 3
Standard cell library, 313
Standard Delay Format (SDF), 232
Static timing verification, 215
Static timing verifier, 323
Stimulus block, 18–19, 20–23
Stimulus, components of, 18–19
$stop task, 292
Strength levels, 32, 363
Strings, 30, 40, 417
$strobe task, 201–2
Strobing, 202–3
Structural coverage, 351–52
Structured procedures, 120–24
always statement, 123–24
initial statement, 120–23
Stuck-at faults, 422
supply1 and **supply0** keywords, 239–40, 366
Switch level, abstraction, 16

Switch-level modeling, 235–50
bidirectional switches, 238–39
CMOS switches, 237–38
delay specification on switches, 241–42
bidirectional pass switches, 242
MOS and CMOS switches, 241
specify blocks, 242
elements, 235–42
examples, 242–48
2-to-1 multiplexer, 245–46
CMOS **nor** gate, 242–45
simple CMOS latch, 246–48
MOS switches, 236–37
power and ground, 239–40
resistive switches, 240–41
Switches, 235
Symbols, 96–97, 135
Synopsys synthesis products, 299
Synthesis design flow, 311–20
RTL to gates, 311–20
design constraints, 314
example of, 315–20
logic optimization, 313
optimized gate-level description, 315
RTL description, 312
technology library, 313–14
technology mapping and optimization, 313
translation, 312–13
unoptimized intermediate representation, 313
Synthesis flow, 315
Synthesizable FIFO model, example, 427–35
System functions, 387
System tasks, 40–44, 199–208, 387
displaying hierarchy, 201–2
displaying information, 41–42
file output, 199–201
closing files, 201
opening a file, 199–200
writing to files, 200–201
initializing memory from file, 204–6
monitoring information, 42–43
random number generation, 203–4
stopping and finishing in a simulation, 43–44
strobing, 202–3
value change dump (VCD) file, 206–8
System timing checks, 409–12
command arguments, 410–11
commands, 409–10
definitions, 411–12

T

T-flipflops (T_FFs), 13–14, 17, 20
Tabs, 27
Target technology, 313
Task declarations, 397
Task enable statements, 406
Task/function routines, 273
task keyword, 172
Tasks, 171–77
automatic (re-entrant), 176–77
compared to functions, 171–72
declaration/invocation, 173–74
endtask keyword, 172
examples, 174–76
asymmetric sequence generator, 175–76
input and output arguments, 174–75
task keyword, 172
when to use, 172
Technology-dependent optimization, 313
Technology-independent design, 304
Technology library, 314
Technology mapping, 313
Technology optimization, 313
Terminals, *See* Ports
Ternary operators, 28
Test bench, 18
$test$plusargs system task keyword, 195
tf routines, *See* Utility routines
time data type, 37
Time scales, 197–98
‘timescale directive, 197–98
Timing and delays, 215–34
delay back-annotation, 230–32
delay models, 216–19
distributed delays, 216–17
lumped delays, 217–18
pin-to-pin delays, 218–19
path delay modeling, 219–27
conditional path delays, 224–25
edge-sensitive paths, 223
fall delays, 225–26
full connection, 222–23
handling **x** transitions, 227
max delays, 225–26
min delays, 225–26
parallel connection, 221
rise delays, 225–26
specify blocks, 219–27
specparam statement, 223–24
turn-off delays, 225–26
typical delays, 226–27
timing checks, 227–30
$hold task, 229
$setup task, 228–29
$swidth task, 230
Timing, as design constraint, 314
Timing checks, 227–30
$hold task, 229
$setup task, 228–29
$width task, 230
Timing control statements, 404–5
Timing controls, 129–36
delay-based timing control, 129–32
intra-assignment delay control, 131–32
regular delay control, 130–31
zero delay control, 132
event-based timing control, 133–36
event OR control, 134–36
named event control, 133–34
regular event control, 133

level-sensitive timing control, 136
Timing simulation, 215, 323
Top-down design methodology, 11–13
tran switch, 238–39, 242
trand keyword, 366–67
tranif0 switch, 238–39, 242
tranif1 switch, 238–39, 242
Transistors, 235
tri keyword, 364–65
tri0 keyword, 366
tri1 keyword, 366
trior keyword, 366–67
trireg keyword, 365
trireg nets, 33
Turn-off delay, 225–26
gates, 79
typ value, 80–81
+typdelays option, 227
Type declarations, 394
Typical delays, 226–27
Typical design flow, 5–6

U

Unary operators, 28, 98
Underscore characters, 30
$ungetc task, 200*fn*
Unoptimized intermediate representation, 313
Unsigned quantities, 36
$unsigned task, 199
Unsized numbers, 29
User-defined primitives (UDPs), 251–70
basics of, 251–53
combinational UDPs, 251, 253–60
definition, 253–55
example of, 258–60
instantiating UDP primitives, 257
shorthand notation for don't cares, 256
state table entries, 255–56
definition, parts of, 252
design guidelines, 267–68
primitive keyword, 252
rules, 253
sequential UDPs, 251
table keyword, 252
UDP body, 402
UDP declaration, 401
UDP instantiation, 402
UDP ports, 401–2
UDP table shorthand symbols, 266–67
User-defined system tasks, 274
Utility access routines, 285, 376–77
Utility routines, 291–95, 378–83
Display Messages, 382
examples of, 292–95
Get Argument List Information, 378
Get Calling Task/Function Information, 378
Get Parameter Values, 379
housekeeping tasks, 383
Long Arithmetic, 382
mechanics of, 291–92
miscellaneous, 383
Monitor Parameter Value Changes, 380
Put Parameter Value, 379
Synchronize Tasks, 380–81
types of, 292

V

Value change dump (VCD) file, 206–8
Value Change Link (VCL) routines, 285, 373
Value levels, 32
Value set, 32–33
$value$plusargs system task keyword, 196
Vectors, 34–35
variable vector part select, 35
vector part select, 35
Verification flow:
analysis, 350–51
architectural modeling, 343–44
block level verification, 345
coverage, 351–52
design-under-test (DUT), 343
extended verification, 345
full chip verification, 345
functional verification environment, 344–47
high-level verification languages (HVLs), 346–47
simulation, 347–50
Verilog coding style, 323–27
basic building blocks vs. continuous **assign** statements, 324–25
defining **if-else** or **case** statements explicitly, 326
mixing positive/negative edge-triggered flip-flops, avoiding, 324
multiple assignments to the same variable, 326
multiplexers vs. **if-else** or **case** statements, 325
parentheses, using to optimize logic structure, 325
signals/variables, using meaningful names for, 324
using arithmetic operators vs. designing building blocks, 325
Verilog HDL, 4
abstraction levels, 8
advanced verification techniques, 341–60
assertion checking, 352–54
formal verification, 355–58
traditional verification flow, 342–52
architectural modeling tools, 423
assertion checking tools, 424
basic concepts, 27–47
compiler directives, 44–45
coverage tools, 424
cycle simulation, 422
data types, 32–40
equivalence checking tools, 424
event-driven simulation, 422
examples, 427–40
behavioral DRAM model, 436–40
synthesizable FIFO model, 427–35

fault simulation, 422
formal verification tools, 424
gate-level modeling, 63–87
general Web sites, 423
hardware acceleration tools, 423
high-level verification languages (HVLs), 423
in-circuit emulation tools, 424
lexical conventions, 27–31
libraries, 8
local parameters, 39
logic synthesis with, 299–340, *See also* Logic synthesis
origins of, 421
popularity of, 7–8
simulation tools, 423
simulators, 421–22
synthesis, 304-11
system tasks, 40–44
Verilog HDL synthesis, 304–11
Verilog constructs, 305–11
assign construct, 307–9
case construct, 309
function statement, 311
if-else construct, 309
for loops, 310–11
Verilog operators, 306–7
Verilog PLI, *See also* Programming Language Interface (PLI)
generations of, 273
Verilog Procedural Interface routines, 273
Vertical partitioning, 328
VHDL, 4
VLSI (Very Large Scale Integration} technology, 3

W

wand keyword, 366–67
Waveform viewers, 351
What-if scenarios, verifying, 304
while loop, 141–43
White box verification approach, 353
Whitespace, 27, 419
$width task, 230
wor keyword, 366–67
Words, 38
$write task, 201–2
Writing to files, 200–201

X

X or Z values, 29