References

Bergeron, Janick. *Writing Testbenches Using SystemVerilog*. Norwell, MA: Springer, 2006

Bergeron, Janick, Cerny, Eduard, Hunter, Alan, and Nightingale, Andrew. *Verification Methodology Manual for SystemVerilog*. Norwell, MA: Springer, 2006

Cohen, Ben, Venkataramanan, Srinivasan, and Kumari, Ajeetha. *SystemVerilog Assertions Handbook for Formal and Dynamic Verification*: VhdlCohen Publishing 2005

Cummings, Cliff. *Nonblocking Assignments in Verilog Synthesis, Coding Styles That Kill!* Synopsys User Group, San Jose, CA, 2000

Cummings, Cliff, Salz, Arturo. *SystemVerilog Event Regions, Race Avoidance & Guidelines*, Synopsys User Group, Boston, CA, 2006

Denning, Peter. *The Locality Principle*, Communications of the ACM, 48(7), July 2005, pp. 19–24

Haque, Faisal, Michelson, Jonathan. *The Art of Verification with SystemVerilog Assertions*. Verification Central 2006

IEEE *IEEE Standard for SystemVerilog — Unified Hardware Design, Specification, and Verification Language*. New York: IEEE 2009 (a.k.a. SystemVerilog Language Reference Manual, or LRM.)

IEEE *IEEE Standard Verilog Hardware Design, Description Language*. New York: IEEE 2001

Rich, Dave *Are SystemVerilog Program Blocks Needed?* http://blogs.men-tor.com/verificationhorizons/blog/2009/05/07/programblocks/ 2009

Sutherland, Stuart. *Integrating SystemC Models with Verilog and SystemVerilog Using the SystemVerilog Direct Programing Interface*. Synopsys User Group Europe, 2004

Sutherland, Stuart, Davidmann, Simon, Flake, Peter, and Moorby, Phil. *System-Verilog for Design: A Guide to Using SystemVerilog for Hardware Design and Modeling*. Norwell, MA: Springer, 2006

Sutherland, Stuart, Mills, Don. *Verilog and SystemVerilog Gotchas*. Norwell, MA: Springer, 2007

Synopsys, Inc., *Hybrid RTL Formal Verification Ensures Early Detection of Corner-Case Bugs*, http://synopsys.com/products/magellan/magellan_wp.html, 2003

van der Schoot, Hans, and Bergeron, Janick *Transaction-Level Functional Coverage in SystemVerilog*. San Jose, CA: DVCon, February 2006

Vijayaraghavan, Srikanth, and Ramanathan, Meyyappan. *A Practical Guide for SystemVerilog Assertions*. Norwell, MA: Springer, 2005

Wachowski Andy, and Wachowski Larry. *The Matrix*. Hollywood, CA: Warner Brothers Studios, 1999

Index

!, 179
--, 69
## cycle delay, 108, 204, 375
$, 37, 343–344
$bits, 305
$cast, 55, 60, 284, 286, 291, 305
$clog2, 28
$dist_exponential, 195
$dist_normal, 195
$dist_poisson, 195
$dist_uniform, 195
$error, 117
$exit, 110–111
$fatal, 117
$fclose, 70
$feof, 40, 70
$finish, 110–111, 426
$fopen, 40, 70
$fscanf, 40, 70–71
$get_coverage, 359
$info, 117
$isunknown, 27, 118
$monitor, 372
$psprintf, 62
$random, 195, 217
$realtime, 82–83
$root, 115
$sformat, 62
$sformatf, 62
$size, 30, 35, 41
$strobe, 372
$system, 452
$test$plusargs, 390–391
$time, 27, 82–83
$timeformat, 27, 82–83
$typename, 305, 310
$unit, 114, 417, 449
$urandom, 195, 202, 251
$urandom_range, 41, 194, 195, 202, 215
$value$plusargs, 311, 390–391, 452
$warning, 117
%0t, 27
%p, 29–30, 41, 309
%t, 82
++, 69
+=, 69–70
+ntb_random_seed, 224, 390
::, 54, 142, 144, 250
<<, 55
<->, 186, 190
-=, 69
->, 184, 188–189, 192
->, 190
>>, 55–57, 161
? : operator, 32, 43
[$], 37
[], 35
^=, 69, 74, 76
__FILE__, 174
__LINE__, 174
`define, 49
‘include, 53, 93
‘SV_RAND_CHECK, 174

Numerics
2-state types, 27, 416, 420, 423, 426
4-state types, 27, 424–426

A
Abstract class, 295–296, 298–299, 311, 314, 315
Accellera, viii
Accessor methods, 162
Active region, 101–103, 106, 112
Agent, 17, 163
AHB, 128
Always block, 230

Always block in programs, 111
and method, 41
Anonymous enumerated type, 57
API *see* Application Programming Interface (API)
Application Programming Interface (API), ix
Arguments
  default value, 75–76
  sticky, 73
  task and function, 72
  type, 73
Array
  assignment, 37
  associative, 38–42, 47
  compare, 32
  constraint, 203
  copy, 32
  dynamic, 35–36, 38, 46, 47
  fixed-size, 28, 36, 38, 46, 47
  handle, 155
  literal, 29, 35, 37, 40
  locator methods, 42
  methods, 41
  multi-dimensional, 28, 30–31, 36, 436
  open, 435–436
  packed, 33–34
  queue, 36, 38, 46, 47
  reduction methods, 41
  slice, 32
  unpacked, 29, 34
Assertion
  concurrent, 118
  coverage, 328
  immediate, 116–117, 286
  in an interface, 379
Assignment pattern, 29
Associative array, 38–42, 47, 50
at_least option, 331, 357
ATM router, 119–121, 123–125
Atomic stimulus generation, 211
auto_bin_max, 339–340, 347, 355
Automatic, 74, 80–81, 103, 137, 138, 141, 237
  in threads, 236
Aval, 426, 428

B
Backdoor load, 115
Base class, 276–277
BaseTr class, 295, 304–305
Begin…end, 72, 230
  optional in tasks and functions, 72
Bergeron, Janick, xiii, 4, 387, 455
BFM *see* Bus Functional Model (BFM)
Bus Functional Model (BFM), 13
Bidirectional constraint, *see* Constraint bidirectional
Bidirectional signal, 108
Binsof, 351, 353
Bit data type, 27, 421, 432
Bit streaming *see* Streaming operator
Blueprint pattern, 279
BNF *see* Backus-Naur Form (BNF)
Backus-Naur Form (BNF), 213
Bounded mailbox, 256–257
Break, 70
Bval, 424, 426
Byte data type, 27, 418, 430

C
calc_csm method, 275–276, 287
Callback
  coverage, 335
  creation, 298
  inject distrubance, 297
  scoreboard, 300
  usage, 299
Case, 71
Cast, 54, 60, 63, 71
  function return value, 71
Cerny, Eduard, 455
Chandle data type, 419, 421–423, 430
Char, 419
Checker, 17, 163
Class, 132, 134
Class constructor, 136
Class scope resolution operator, 144, 250
Clock generator, 111–112
Clocking block, 98–99, 103, 108, 109, 123, 124, 365
Cohen, Ben, 455
Command layer, 17
Comment
  covergroup, 356
Comment option, 333, 356
Compilation unit, 113
Composition, 273, 288, 290, 292
Compound assignment, 69–70, 74
Concatenation
  bits, 56
  string, 61
Concurrent assertion, 118
Conditional operator, 32, 43
Config class, 266
Configuration database, 308
Const, 76
Const type, 61, 74, 114, 176

Constrained, 7
Constraint
  array, 203
  bidirectional, 184, 188
  block, 173
  dist, 178–179, 196, 359
  implication, 188
  in extended class, 284
  inside, 176, 179, 183, 202, 203
  solve…before, 190–191, 359, 396
constraint_mode, 191–193, 280
Constructor, 136–137, 277, 295
Consumer, 244
Containment, 149
Context, 444
Continue, 70
Copy
  deep, 159
  method, 159–160, 280, 293, 294
  object, 156, 293
  shallow, 158
Coverage event, 335
Covergroup
  comment, 356
  embedded, 334
  generic, 354–355
  option, 355
  sample, 331, 335–337
  trigger, 335
Coverpoint, 330, 334, 337, 340, 343, 351, 353, 358
Cross coverage, 348–354
Cross module reference, 114, 370, 376
cross_num_print_missing option, 357
CRT *see* constrained-random test (CRT)
Constrained-Random Test (CRT), 4, 7–8, 169, 196, 323
Cummings, Cliff, xiii, 100, 102, 455
Cyclic random, 173

D
Data type
  bit, 26–27
  byte, 26–27
  chandle, 419, 421–423, 430
  int, 27
  integer, 25
  logic, 26
  longint, 27
  real, 25
  reg, 25–26
  shortint, 27
  time, 25
  wire, 26
  Deallocation, 139
DDR *see* Double data rate (DDR)
Decrement, 69
Default clocking block, 108
Default coverage value, 343, 345, 350–351
Default statement, 98
Default value, 75–76
Delete method, 35, 37, 39, 48
Derived class, 277
Design pattern, 314
Design Under Test (DUT), 3, 88
Direct Programming Interface (DPI), 415–454
Disable, 231, 240–243
Disable fork, 241–242
Disable label, 242–243
Display method, 141–142, 275, 276, 285
Dist constraint, 178, 179, 196, 359
do…while, 39, 59, 70, 174
Domain, 338–339
Double data rate (DDR) clock, 98
Double data type, 418–419
Double precision floating point, 28
Downcasting, 284
DPI *see* Direct Programming Interface (DPI)
DPI-C, 416, 423
Driver, 17, 163
Driver class, 375, 379
DUT *see* Design Under Test (DUT)
Dynamic array, 35–36, 38, 46, 47
Dynamic cast, *see* $cast
Dynamic threads, 235

E
e Reuse Methodology (eRM), viii
Embedded covergroup, 334
Enum, 57–60
Enumerated types, *see* enum
Enumerated values, 58
Enumeration, 57
Environment, 18, 163
Environment class, 163
Equivalence operator, 186, 190
eRM *see* e Reuse Methodology (eRM)
Event, 244–249, 264
Event triggered, 244–249, 261
Exists, 40, 48
Expression width, 62
Extended class, 276–277
Extern, *see* External routine declaration
External constraint, 199–200
External routine declaration, 142–143, 266, 296

F
Fabs routine, 417
File I/O, 70
File_exists method, 417
Final block, 110
Find_first method, 43
Find_index method, 43–45
Find_last method, 43
Find_last_index method, 43
First, 39
First method, 39, 47, 59–60
Fixed-size array, 28, 35, 38, 46, 47
Float data type, 418
for loop, 30, 39, 59, 69, 197
Force design signals, 115
foreach constraint, 203, 207–209, 396
foreach loop, 30–31, 35, 37, 40
Fork…join, 230–231
Fork…join_any, 230, 233
Fork…join_none, 230, 232, 234, 235, 237, 238
Four-state types, *see* 4-state types
Function, 71
    arguments, 72
Functional coverage, 18, 455
    using callbacks, 301
Functional layer, 17

G
Garbage collection, 139–140
Generator, 18, 163
Generator class, 234, 248, 250, 255, 279, 281, 304, 305
Get method, 251–254, 261, 263, 265
get_coverage, 359
get_inst_coverage, 359
Getc method, 61–62
Goal option, 358

H
Handle, 134–135, 139
    array, 155
Haque, Faisal, 455
Hardware Description Language (HDL), 2, 83
Hardware Verification Language (HVL), viii, 2, 13
HDL *see* Hardware Description Language (HDL)
Histogram, 180
Hook, 280, 297
HVL *see* Hardware Verification Language (HVL)

I
IEEE 1800, see LRM
If constraint, 185–186
Iff, 118, 344
Ignore_bins, 346–347, 351
Illegal_bins, 347
Immediate assertion, 116–117, 286
Implication, 184
Implication operator, 184, 190
Implicit port connection, 112
Import, 53, 418–419
Incl.h, 418
Increment, 69
Inheritance, 274, 288
Initial, 29
Initialization in declaration, 80
In-line constraint, 199
Inout
    argument type, 73
    port type, 126
Input
    argument type, 73
    port type, 126
Insert queue, 37
Inside, 71, 176, 180
Inside constraint, 176, 179, 183, 202, 203
Instance, 136
Instantiation, 136
Int data type, 27, 418, 419, 430
Integer data type, 25, 27, 51
Interface, 91, 97, 113, 369
    connecting to port, 93
    procedural code, 379
    virtual, 363–364, 366, 368, 370, 379
Interprocess, 2
Interprocess communication (IPC), 229, 244, 264
Intersect, 351, 353
io_printf, 422, 436
IPC *see* Interprocess Communication (IPC)
Iterator argument, 43

L
Language Reference Manual (LRM), vii–viii, 47, 76, 100, 102, 109, 175, 200, 229, 333, 352, 363
Last method, 59
Local, 162, 302, 303, 308
Logic, 49, 91
Logic data type, 26, 27, 97, 419, 430
Long long int, 437
Longint data type, 26, 418, 419, 430
LRM *see* Language Reference Manual (LRM)

M
Macro, 48
Macromodule, 114
Magellan, 455
Mailbox, 252, 257, 262–264
  bounded, 256, 257
  unbounded, 256
Makes Jack a dull boy
Malloc, 136, 422
Max method, 42
Method, 134, 277
  virtual, 286–288, 293, 294
Min method, 42
Modport, 94, 123
Module, 133
Monitor, 17, 163
Multi-dimensional array, 28, 30–31, 36, 436

N
Name function, 182
Name method, 57
New
  constructor, 136–137
  copying objects, 156
New function, 137
New[] operator, 35, 138
Next method, 39, 48, 59
Nonblocking assignment, 92, 98, 100, 101, 106
Null, 136, 139, 140, 198
Num method, 39, 255

O
Object, 134, 135
  copy, 156, 293
  deallocation, 139
Object-Oriented Programming (OOP), vii, 131, 132, 273
  analogy badge, 139
  analogy car, 132
  analogy house, 135
  terminology, 134, 277
Observed region, 102
OOP, *see* Object-Oriented Programming (OOP)
Open array, 434
Open array, *see* array open
Open Verification Methodology (OVM), viii, 4, 309
OpenVera, viii
Option, 352
  at_least, 331, 357
  auto_bin_max, 339, 340, 347, 355
  comment, 333, 356
  cross_num_print_missing, 357
  goal, 358
  per_instance, 356, 359
  weight, 351–353, 355
Output
  argument type, 73
  port type, 126
OVM, *see* Open Verification Methodology (OVM)

P
Pack
  method, 160, 161, 438–439
  operation, 55
Package, 49, 53, 61, 114, 133, 146, 200
Packed array, 33–34
Packed structure, 52
Parameter, 49, 61, 114
Parameterized
  class, 302
  interface, 377
  mailbox, 253
  module, 377
  virtual interface, 377–379
Parent class, 277
Peek method, 253
Per_instance option, 356, 359
Perl hash array, 39
Physical interfaces, 363
PLI, 415, 422
Plyant, Tim
  a good guy
Polymorphism, 287–288
Pop_back method, 37
Pop_front method, 37
Port
  connecting to interface, 93
Post_randomize, 193–195, 210, 212
Post-decrement, 69
Post-increment, 69
Postponed region, 102–105
Pre_randomize, 193–195, 202
Pre-decrement, 69
Pre-increment, 69
Prev method, 47, 59
Primitive, 114
Print class, 306
Pseudo-random number generator (PRNG), 175, 217, 219
Process, 229
Producer, 244

Product method, 41
Program, 80, 103, 114, 133, 146, 238, 267–268
Program blocks
  single vs. multiple, 103
Property, 136, 277
Protected, 308
Prototype, 134, 277, 295
Proxy class, 314–315
PRNG *see* Pseudo-random number generator(PRNG)
Public, 162
Pure
  imported method, 440–441
  virtual method, 295–296, 299, 311
  virtual methods, 315
Push_back method, 43, 299
Push_front, 37
Push_front method, 37, 299
Put method, 251, 253, 254, 257–258, 261, 263, 265
Putc method, 62

Q
Queue, 36, 38, 46–47
  literal, 37–38

R
Ramanathan, Meyyappan, 455
Rand, 173
Rand_mode, 191, 197
Randc, 173, 182, 203, 208, 281
Randcase, 215–216
Random seed, 11–12, 170, 175, 217, 219, 224
Random stability, 217, 219
Randomize function, 173, 176
Randomize() with, 192–193, 199
Randomize(null), 198
Reactive region, 102–103, 106
Real data type, 25, 27, 175, 418, 430
Realtime data type, 82–83
Ref
  argument type, 76, 418–419
  port type, 126
Reg data type, 25–26
Regions
  scheduling, 101, 102
Return, 78
Reverse method, 45
Rich, Dave, 455
Routine arguments, 72
Run method, 279, 281–282
RVM, 4

S
S_vpi_vecval, 424
Sample method, 331, 335–337
Scenario generation, 211–212
Scenario layer, 18
Scheduling region, 101
Scheduling regions, 102
Scope operator, 54, 142
Scoreboard, 18, 45–46, 163, 297, 300–301
  using callbacks, 297
seed, *see* Random seed
Semaphore, 250–252, 264
Shortint data type, 29, 418, 430
Shortreal data type, 418, 430
Shuffle method, 45
Signal layer, 17
Signature, 288
Signed data types, 27
Sine function, 441
Singleton class, 306, 308, 315
Size function, 203
Size method, 35, 39
Solve…before, *see* Constraint solve…before
Sort method, 45
Sparse matrix, 38
Specialization, 303, 305
Start method, 344
Stat method, 417
State variables, 198
Static, 29
  cast, 54
  class, 306, 311
  method, 145, 311
  storage, 80
  variable, 143–146, 311
Stop method, 344
Streaming operator, 55–56, 160–161
String concatenation, 61
String data type, 61–62, 419, 430
Stringify, 174, 317
Struct, 50–53, 57
Subclass, 277
Substr method, 62
Sum, 55
Sum method, 41, 43–44, 204–207, 396
Superclass, 277
Sutherland, Stuart, 385
SV_PACKED_DATA_NELEMS, 424

SV_RAND_CHECK, 174, 286
SVA, 118, 328, 338
SvBit, 419–420, 422, 427–428
SvBitVecVal, 419–420, 423, 426, 435
SvDimension, 435
Svdpi.h, 418, 420, 422, 435
SvGetArrayElemPtr1, 437
SvGetArrayPtr, 434, 435
SvGetArrElemPtr, 435
SvGetArrElemPtr1, 435
SvGetArrElemPtr2, 435–436
SvGetArrElemPtr3, 435
SvGetNameFromScope, 450
SvGetScope, 450
SvGetScopeFromName, 450
SvHigh, 435–436
SvIncrement, 435
SvLeft, 435–436
SvLogic, 418, 426
SvLogicVecVal, 419, 424–426, 435
SvLow, 435–436
SVM, 314
SvOpenArrayHandle, 434, 435
SvRight, 435–436
SvScope, 450
-Svseed, 224
SvSetScope, 450
SvSize, 435
SvSizeOfArray, 435
Synchronous drive, 106
System(), 452
SystemC, 415
SystemVerilog, *see* LRM
SystemVerilog Assertion, *see* SVA
SystemVerilog Methodology, 314

T
Task, 71
  Arguments, 72
Template, *see* Parameterized class
Test class, 311
Test layer, 18
The Matrix, 1
This, 148–149
Thread, 229, 232–236, 238–240
Time data type, 25, 27, 82
Time literals, 82
Time step, 102
Timeprecision, 82
Timescale, 81–83, 114
Timeunit, 82
TLM analysis port, 302
Tolower method, 61
Toupper method, 61–62
Transactor, 163–164
Transition coverage, 345–346
Tri data type, 26
Triggered, 246
Triggered method, 244–249, 261
Try_get method, 251, 255
Try_put method, 253
Two-state types, *see* 2-State types
Type conversion, 54
Type_option, 352
Typedef, 48–49
  array, 50, 79
  associative array index, 50
  class, 152
  enum, 58
  struct, 51–52, 57
  union, 52
  virtual interface, 375

U
Uint user-defined type, 49, 206
Unbounded mailbox, 256
Union, 51–52
Unique method, 42
Universal Verification Methodology (UVM),
    viii, 4–5, 20, 160, 212, 279, 299,
    305, 309, 314, 336
Unpack
  method, 160–161, 438–439
  operation, 55
Unpacked array, 29, 33–34
Unsigned, 27, 49, 195, 201, 208
Unsized array, *see* array open
User defined type, *see* typedef
UVM *see* Universal Verification Methodology
    (UVM)

V
Van der Schoot, Hans, 455
Vc_hdrs.h, 418
Verification IP, *see* VIP
Verification Methodology Manual for
    SystemVerilog, *see* VMM
Verification Procedural Interface,
    see VPI
Verilog-1995, vii–viii, 25, 28, 32, 72–73, 80,
    119, 195, 363, 377
Verilog-2001, vii–viii, 26, 28, 32, 33, 61, 62,
    70, 72, 80, 89, 93
Veriuser.h, 422
Vijayaraghavan, Srikanth, 455

VIP, 14
Virtual
    class, *see* Abstract classinterface, 309, 363–379
    memory, 287
    method, 279, 286–288, 293–295
VMM, viii, 4–5, 17, 20, 97, 212–213, 279, 299, 307, 314, 328, 371
Void
    data type, 71, 419, 430
    function, 71, 195
VPI, 415, 424, 426
Vpi_control, 426
VpiFinish, 426

W
Wait, 231, 235, 241, 245–250, 268
Wait fork, 238–239, 248
Weight option, 351–353, 355
Wildcard, 346, 354
Wire data type, 26, 97
With, 42–44
Wrap_up method, 282

X
XMR, 370–371, 376
Xor method, 41