

Contents

1	Introduction	1
1.1	SystemVerilog Language Evolution	2
2	Data Types	5
2.1	Integer Data Types	5
2.1.1	Integer, int, longint, shortint, logic, byte, reg.	6
2.1.2	Signed Types	8
2.1.3	Bits vs. Bytes	10
2.2	Real Data Types	10
2.2.1	“real” Data-Type Conversion Functions	11
2.3	Nets	12
2.3.1	“wire” and “tri”	13
2.3.2	Unresolved “wire” Type: “uwire”	15
2.3.3	Resolved vs. Unresolved Type	16
2.3.4	“wand” and “triand”	17
2.3.5	“wor” and “trior”	17
2.3.6	“tri0” and “tri1”	17
2.4	Drive Strengths	19
2.5	Variable vs. Net	22
2.6	“var”	23
2.7	Variable and Net Initialization	23
2.8	Static, Automatic, and Local Variables	25
2.8.1	Static vs. Local Variables	25
2.8.2	Automatic vs. Static Variable	27
2.8.3	Variable Lifetimes	29
2.9	Enumerated Types	30
2.9.1	Enumerated-Type Methods	32
2.9.2	Enumerated Type with Ranges	36
2.10	User-Defined Type: Typedef	38

2.11	String Data Type	40	5.4	Structure as an Argument to Task or Function.....	130
2.11.1	String Operators.....	41	5.5	Structure Within a Structure.....	131
2.11.2	String Methods.....	44			
2.12	Event Data Type.....	47	6	Union	133
2.12.1	Event Sequencing: wait_order ()	50	6.1	Packed and Unpacked Unions	135
2.13	Static Casting.....	51	6.1.1	Unpacked Unions.....	136
2.13.1	Bit-Stream Casting.....	56	6.1.2	Tagged Unions	140
2.14	Dynamic Casting	57	6.1.3	Packed Union.....	142
3	Arrays	61	7	Packages	145
3.1	Packed and Unpacked Arrays	61	8	Class	155
3.1.1	2-D Packed Array.....	62	8.1	Basics.....	155
3.1.2	3-D Packed Array.....	63	8.2	Base Class	156
3.1.3	1-D Packed and 1-D Unpacked Array	65	8.3	Extended Class and Inheritance.....	162
3.1.4	4-D Unpacked Array	67	8.3.1	Inheritance Memory Allocation.....	167
3.1.5	1-D Packed and 3-D Unpacked Array	68	8.4	Class Constructor.....	169
3.1.6	2-D Packed and 2D-Unpacked Array.....	69	8.4.1	Base Class Constructor	169
3.1.7	3-D Packed and 1-D Unpacked Array	71	8.4.2	Chain Constructor (super.new()).....	169
3.2	Assigning, Indexing, and Slicing of Arrays.....	72	8.5	Static Properties	172
3.2.1	Packed and Unpacked Arrays as Arguments to Subroutines	74	8.6	Static Methods	176
3.3	Dynamic Arrays.....	74	8.7	"this".....	180
3.3.1	Dynamic Arrays – Resizing.....	75	8.8	Class Assignment.....	184
3.3.2	Copying of Dynamic Arrays	79	8.9	Shallow Copy.....	186
3.3.3	Dynamic Array of Arrays.....	80	8.10	Deep Copy	189
3.4	Associative Arrays	83	8.11	Upcasting and Downcasting.....	194
3.4.1	Wild Card Index.....	84	8.12	Polymorphism	198
3.4.2	String Index	85	8.13	Virtual Methods, Pure Virtual Methods, and Virtual Classes.....	203
3.4.3	Class Index.....	85	8.13.1	Virtual Methods	203
3.4.4	String Index – Example	86	8.13.2	Virtual (Abstract) Class and Pure Virtual Method.....	205
3.4.5	Associative Array Methods	87	8.14	Data Hiding ("local," "protected," "const") and Encapsulation.	211
3.4.6	Associative Array – Default Value.....	90	8.14.1	Local Members	212
3.4.7	Creating a Dynamic Array of Associative Arrays	91	8.14.2	Protected Members	215
3.5	Array Manipulation Methods.....	92	8.14.3	"const" Class Properties.....	217
3.5.1	Array Locator Methods	92	8.15	Class Scope Resolution Operator (::) and "extern"	221
3.5.2	Array Ordering Methods	99	8.16	Parameterized Class.....	225
3.5.3	Array Reduction Methods	100	8.16.1	Value Parameters	226
4	Queues	105	8.16.2	Type Parameters.....	227
4.1	Queue Methods	109	8.16.3	Parameterized Class with Static Properties	229
4.2	Queue of SystemVerilog Classes.....	117	8.16.4	Extending Parameterized Class	232
4.3	Queue of Queues: Dynamic Array of Queues	118	8.17	Difference Between Class and Struct.....	233
5	Structures	121	9	SystemVerilog "module"	235
5.1	Packed Structure	122	9.1	Module Header Definition.....	236
5.2	Unpacked Structure	126	9.1.1	ANSI-Style Module Header.....	237
5.3	Structure as Module I/O.....	128	9.1.2	ANSI-Style <i>first</i> Port Rules	239
			9.1.3	ANSI-Style <i>subsequent</i> Port Rules	240

9.1.4	Non-ANSI-Style Module Header	241
9.2	Default Port Values	245
9.3	Module Instantiation	247
9.3.1	\$root	249
9.4	Nested Modules	249
9.5	Module Parameters	250
9.5.1	Overriding Module Parameters: <i>defparam</i>	251
9.5.2	Overriding Module Parameters: <i>Module Instance Parameter Value Assignment</i>	252
9.5.3	Localparam	253
9.5.4	Parameter Dependence	254
10	SystemVerilog “program”	257
10.1	Eliminating Testbench Races	261
11	SystemVerilog “interface”	267
11.1	Interfaces	267
11.2	Modports	274
11.3	Tasks and Functions in an Interface	278
11.3.1	“import” Tasks in a Modport	280
11.3.2	“export” Tasks in a Modport	282
11.4	Parameterized Interface	283
11.5	Clocking Block in an Interface	286
12	Operators	289
12.1	Assignment Operators	289
12.2	Increment and Decrement Operators	290
12.3	Arithmetic Operators	292
12.4	Relational Operators	296
12.5	Equality Operators	298
12.5.1	Wildcard Equality Operators	300
12.6	Logical Operators	301
12.7	Bitwise Operators	303
12.8	Unary Reduction Operators	305
12.9	Shift Operators	307
12.10	Conditional Operators	308
12.11	Concatenation Operators	310
12.12	Replication Operators	311
12.13	Streaming Operators (pack/unpack)	312
12.13.1	Packing of Bits	314
12.13.2	Unpacking of Bits	318
12.14	“inside” Operator (Set Membership Operator)	321
13	Constrained Random Test Generation and Verification	325
13.1	Productivity Gain with CRV	326
13.2	Constrained Random Verification (CRV) Methodology	327
13.3	SystemVerilog Support for CRV	328

13.4	Constraints	329
13.4.1	Constraints: Turning On and OFF	333
13.5	Random Variables (rand and randc)	337
13.5.1	Static Random Variables	339
13.5.2	Randomizing Arrays and Queues	340
13.5.3	Randomizing Object Handles	342
13.6	Constraint Blocks	343
13.6.1	External Constraint Blocks	343
13.6.2	Weighted Distribution	345
13.6.3	“unique” Constraint	348
13.6.4	Implication and If-Else	351
13.6.5	Iterative Constraint (<i>foreach</i>)	352
13.6.6	Array Reduction Methods for Constraint	355
13.6.7	Functions in Constraints	356
13.6.8	Soft Constraints	357
13.7	Randomization Methods	361
13.7.1	Pre-randomization and Post-randomization	362
13.7.2	Local Scope Resolution (local::)	367
13.8	rand_mode(): Disabling Random Variables	367
13.9	constraint_mode(): Control Constraints	370
13.10	randomize() with Arguments: In-Line Random Variable Control	372
13.11	Random Number Generation System Functions and Methods	376
13.11.1	Random Number Generator (RNG)	376
13.11.2	\$urandom() and \$urandom_range()	377
13.11.3	srandom(), get_randstate(), and set_randstate()	380
13.12	Random Stability	385
13.13	Randcase	389
13.14	randsequence	391
13.14.1	Random Production Weights and If-Else Statement	393
13.14.2	Repeat Production Statement	395
13.14.3	rand join	397
13.14.4	break and return	401
13.14.5	Passing Values Between Productions	404
14	SystemVerilog Assertions	409
14.1	SystemVerilog Assertions Evolution	410
14.2	What Is an Assertion?	410
14.3	Why Assertions? What Are the Advantages?	411
14.3.1	Assertions Shorten Time to Develop	412
14.3.2	Assertions Improve Observability	413
14.3.3	Other Major Benefits	414
14.3.4	One-Time Effort, Many Benefits	415
14.4	Assertions in Static Formal	416

14.5	Methodology Components	417
14.5.1	Types of Assertions to Add	417
14.5.2	How to Add Assertions? What Is the Protocol?	418
14.5.3	How Do I Know I Have Enough Assertions?	419
14.5.4	A Simple PCI Read Example: Creating an Assertion Test Plan	420
14.6	Assertion Types	422
14.7	Conventions Used in This Chapter.	423
14.8	Immediate Assertions.	423
14.8.1	Deferred Immediate Assertions	426
14.9	Concurrent Assertions: Basics	429
14.9.1	Implication Operator	434
14.10	Clocking Basics	436
14.10.1	Default Clocking Block	438
14.10.2	Sampling Edge (Clock Edge).	442
14.10.3	Active Region.	443
14.10.4	Observed Region	443
14.10.5	Reactive Region	443
14.10.6	Preponed Region	443
14.11	Concurrent Assertions Are Multi-threaded	447
14.12	Formal Arguments	448
14.13	Disable (Property) Operator: disable iff.	449
14.14	Severity Levels.	451
14.15	Binding Properties	452
14.15.1	Binding Properties (Scope Visibility)	453
14.15.2	VHDL DUT Binding with SystemVerilog Assertions	455
14.16	Difference Between “sequence” and “property”	456
14.17	Sampled Value Functions.	457
14.17.1	\$rose: Edge Detection in Property/Sequence.	458
14.17.2	\$fell: Edge Detection in Property/Sequence	459
14.17.3	Edge Detection Is Useful Because.	459
14.17.4	\$stable	460
14.17.5	\$past.	461
14.18	Operators	466
14.18.1	##m: Clock Delay	466
14.18.2	##[m:n]: Clock Delay Range	468
14.18.3	[*m]: Consecutive Repetition Operator	474
14.18.4	[*m:n]: Consecutive Repetition Range Operator.	476
14.18.5	[=m]: Non-consecutive Repetition.	482
14.18.6	[=m:n]: Non-consecutive Repetition Range Operator	484
14.18.7	[->] Non-consecutive GoTo Repetition Operator	486
14.18.8	Difference Between [=m:n] and [->m:n]	487
14.18.9	Application: GoTo Repetition – Non-consecutive Operator	489

14.18.10	Sig1 <i>throughout</i> Seq1.	489
14.18.11	Seq1 <i>within</i> Seq2	492
14.18.12	Application: Seq1 “within” Seq2.	492
14.18.13	Seq1 <i>and</i> Seq2	495
14.18.14	Application: “and” Operator	498
14.18.15	Seq1 <i>or</i> Seq2	498
14.18.16	Seq1 “intersect” Seq2	500
14.18.17	Application: “intersect” Operator	501
14.18.18	<i>first_match</i>	505
14.18.19	Application: <i>first_match</i>	505
14.18.20	<i>not</i> Operator	508
14.18.21	<i>if</i> (Expression) property_expr1 <i>Else</i> property_expr2.	509
14.18.22	“iff” and “implies”	509
14.19	System Functions and Tasks	510
14.19.1	\$onehot and \$onehot0	510
14.19.2	\$isunknown	511
14.19.3	\$countones	511
14.19.4	\$countbits.	512
14.19.5	\$assertoff, \$asserton, and \$assertkill	514
14.20	Multiply Clocked Sequences and Properties	514
14.20.1	Multiply Clocked Sequences	515
14.20.2	Multiply Clocked Properties: “and” Operator	516
14.20.3	Multiply Clocked Properties: “or” Operator	517
14.20.4	Multiply Clocked Properties – “not”: Operator	517
14.20.5	Multiply Clocked Properties: Clock Resolution	519
14.20.6	Multiply Clocked Properties: Legal and Illegal Conditions	519
14.21	Local Variables.	520
14.21.1	Application: Local Variables	524
14.22	End Point of a Sequence (.triggered).	525
14.22.1	End Point of a Sequence (.matched)	528
14.23	“expect”	530
14.24	IEEE 1800-2012/2017 Features.	532
14.24.1	\$changed	532
14.24.2	Future Global Clocking Sampled Value Functions	532
14.24.3	past Global Clocking Sampled Value Functions	533
14.24.4	“followed by” Properties #-# and #=#	534
14.24.5	“always” and “s_always” Property	536
14.24.6	“eventually” and “s_eventually”	538
14.24.7	“until,” “s_until,” “until_with,” and “s_until_with”.	539
14.24.8	“nexttime” and “s_nexttime”	541
14.25	Abort Properties: reject_on, accept_on, sync_reject_on, and sync_accept_on	545
14.26	\$assertpassoff, \$assertpasson, \$assertfailoff, \$assertfailon, \$assertnonvacuouson, and \$assertvacuousoff	548

14.27	Embedding Concurrent Assertions in Procedural Block	549
14.28	Nested Implications	552
15	SystemVerilog Functional Coverage	553
15.1	Difference Between Code Coverage and Functional Coverage...	554
15.1.1	Code Coverage	554
15.1.2	Functional Coverage	556
15.2	Functional Coverage Methodology	557
15.3	Covergroup: Basics	558
15.4	Coverpoint: Basics	560
15.4.1	Covergroup/Coverpoint Example	562
15.4.2	Coverpoint Using a Function or an Expression	563
15.5	"bins": Basics	564
15.5.1	Covergroup/Coverpoint Example with "bins"	566
15.5.2	"bins" Filtering	567
15.6	Covergroup: Formal and Actual Arguments	568
15.7	SystemVerilog Class-Based Coverage	569
15.7.1	Class: Embedded Covergroup in a Class	569
15.7.2	Multiple Covergroups in a Class	570
15.7.3	Overriding Covergroups in a Class	570
15.7.4	Parameterizing Coverpoints in a Class	574
15.8	"cross" Coverage	575
15.9	"bins" for Transition Coverage	581
15.10	"wildcard bins"	583
15.11	"ignore_bins"	583
15.12	"illegal_bins"	588
15.13	"binsof" and "intersect"	588
15.14	User-Defined "sample()" Method	591
15.15	Querying for Coverage	596
15.16	Strobe () Method	597
15.17	Coverage Options: Instance-Specific Example (Fig. 15.18)	597
15.18	Coverage Options for "covergroup" Type: Example (Fig. 15.19)	598
15.19	Coverage Options: Instance-Specific Per-syntactic Level	598
15.20	Coverage System Tasks, Functions, and Methods	598
16	SystemVerilog Processes	601
16.1	"initial" and "always" Procedural Blocks	601
16.1.1	"initial" Procedural Block	602
16.1.2	"always" Procedural Block: General	603
16.1.3	"always_comb"	606
16.1.4	always @ (*)	609
16.1.5	"always_latch"	610
16.1.6	"always_ff"	612
16.2	"final" procedure	614

16.3	Parallel Blocks: fork-join	615
16.3.1	fork-join	615
16.3.2	fork-join_any	617
16.3.3	fork-join_none	619
16.4	Level-Sensitive Time Control	620
16.4.1	Wait Fork	622
16.5	Named Event Time Control	624
16.5.1	Merging of Named Events	625
16.5.2	Event Comparison	627
16.6	Conditional Event Control	627
16.7	Disable Statement	628
17	Procedural Programming Statements	631
17.1	if-else-if Statements	631
17.1.1	unique-if and unique0-if	633
17.1.2	priority-if	636
17.2	Case Statements	637
17.2.1	casex, casez, and do not care	640
17.2.2	Constant Expression in "case" Statement	644
17.2.3	One-Hot State Machine Using Constant Case Expression	646
17.2.4	unique-case, unique0-case, and priority-case	648
17.3	Loop Statements	652
17.3.1	The "for" Loop	652
17.3.2	The "repeat" Loop	654
17.3.3	The "foreach" Loop	655
17.3.4	The "while" Loop	658
17.3.5	The "do - while" Loop	659
17.3.6	The "forever" Loop	660
17.4	Jump Statements	662
17.4.1	"break" and "continue"	662
18	Inter-process Synchronization. Semaphores and Mailboxes	665
18.1	Semaphores	665
18.2	Mailboxes	668
18.2.1	Parameterized Mailbox	675
19	Clocking Blocks	677
19.1	Clocking Blocks with Interfaces	683
19.2	Global Clocking	685
20	Checkers	689
20.1	Nested Checkers	694
20.2	Checkers: Legal Conditions	695
20.3	Checkers: Illegal Conditions	696
20.4	Checkers: Important Points	698
20.5	Checkers: Instantiation Rules	701

20.6	Checkers: Rules for “formal” and “actual” Arguments	703
20.7	Checkers: In a Package	704
21	“let” Declarations	705
21.1	“let”: Local Scope	706
21.2	“let”: With Parameters	707
21.3	“let”: In Immediate and Concurrent Assertions	709
22	Tasks and Functions	715
22.1	Tasks	716
22.1.1	Static and Automatic Tasks	718
22.2	Functions	724
22.2.1	Function Called as a Statement	725
22.2.2	Function Name or “return” to Return a Value	726
22.2.3	Void Functions	727
22.2.4	Static and Automatic Functions	728
22.3	Passing Arguments by Value or Reference to Tasks and Functions	730
22.3.1	Pass by Value	730
22.3.2	Pass by Reference	732
22.4	Default Argument Values	735
22.5	Argument Binding by Name	737
22.6	Parameterized Tasks and Functions	738
23	Procedural and Continuous Assignments	745
23.1	Procedural Assignments	746
23.1.1	Blocking Versus Non-Blocking Procedural Assignments	746
23.1.2	Continuous Assignment	751
23.2	Procedural Continuous Assignment: Assign and Deassign	753
23.3	Procedural Continuous Assignment: Force-Release	755
24	Utility System Tasks and Functions	759
24.1	Simulation Control System Tasks	759
24.2	Simulation Time System Functions	760
24.3	Timescale System Tasks	761
24.3.1	\$sprinttimescale	761
24.3.2	\$timeformat	761
24.4	Conversion Functions	763
24.4.1	Conversion to/from Signed/Unsigned Expression	765
24.5	\$bits: Expression Size System Function	766
24.6	Array Querying System Functions	766
24.7	Math Functions	769
24.7.1	Integer Math Functions	769
24.7.2	Real Math Functions	769
24.8	Bit-Vector System Functions	771
24.9	Severity System Tasks	773

24.10	\$random and Probabilistic Distribution Functions	773
24.10.1	\$random	773
24.10.2	Probabilistic Distribution Functions	774
24.11	Queue Management Stochastic Analysis Tasks	775
24.11.1	\$q_initialize	776
24.11.2	\$q_add	776
24.11.3	\$q_remove	776
24.11.4	\$q_full	777
24.11.5	\$q_exam	777
24.11.6	Example of Queue Management Stochastic Analysis Tasks and Functions	778
25	I/O System Tasks and Functions	785
25.1	Display Tasks	785
25.2	Escape Identifiers	788
25.3	Format Specifications	789
25.4	File I/O System Tasks and Functions	792
25.4.1	\$fopen and \$fclose	792
25.4.2	\$fdisplay, \$fwrite, \$fmonitor, and \$fstrobe	797
25.4.3	\$swrite and \$sformat	797
25.5	Reading Data from a File	800
25.5.1	\$fgetc, \$ungetc, and \$fgets	800
25.5.2	\$fscanf and \$sscanf	802
25.5.3	\$ftell, \$fseek, and \$rewind	808
25.5.4	\$fread	808
25.5.5	\$readmemb and \$readmemh	809
25.5.6	\$writememb and \$writememh	813
25.6	\$test\$plusargs and \$value\$plusargs	814
25.6.1	\$test\$plusargs	815
25.6.2	\$value\$plusargs	816
25.7	Value Change Dump (VCD) File	817
25.7.1	\$dumpfile	817
25.7.2	\$dumpvars	818
25.7.3	\$dumpon/\$dumpoff	819
25.7.4	\$dumplimit	819
25.7.5	\$dumpflush	819
25.7.6	\$dumpall	819
25.7.7	\$dumpports	820
26	GENERATE Constructs	821
26.1	Generate: Loop Constructs	822
26.2	Generate : Conditional Construct	825
27	Compiler Directives	831
27.1	`define	831
27.1.1	`undef and `undefineall	836
27.2	`ifdef, `else, `elsif, `endif, and `ifndef	837

27.3	`timescale	839
27.4	`default_nettype	842
27.5	`resetall.	842
Bibliography		843
Index.		845