

Contents

1	Verification Guidelines	1
1.1	The Verification Process	2
1.1.1	Testing at Different Levels	3
1.1.2	The Verification Plan.....	4
1.2	The Verification Methodology Manual.....	4
1.3	Basic Testbench Functionality	5
1.4	Directed Testing.....	5
1.5	Methodology Basics	6
1.6	Constrained-Random Stimulus.....	8
1.7	What Should You Randomize?.....	9
1.7.1	Device and Environment Configuration.....	9
1.7.2	Input Data.....	10
1.7.3	Protocol Exceptions, Errors, and Violations	10
1.7.4	Delays and Synchronization.....	11
1.7.5	Parallel Random Testing	11
1.8	Functional Coverage	12
1.8.1	Feedback from Functional Coverage to Stimulus.....	12
1.9	Testbench Components	13
1.10	Layered Testbench	14
1.10.1	A Flat Testbench	14
1.10.2	The Signal and Command Layers.....	17
1.10.3	The Functional Layer	17
1.10.4	The Scenario Layer	18
1.10.5	The Test Layer and Functional Coverage	18
1.11	Building a Layered Testbench.....	19
1.11.1	Creating a Simple Driver.....	20
1.12	Simulation Environment Phases.....	20
1.13	Maximum Code Reuse.....	21
1.14	Testbench Performance	22
1.15	Conclusion.....	22
1.16	Exercises	23

2 Data Types	25
2.1 Built-In Data Types	25
2.1.1 The <code>Logic</code> Type	26
2.1.2 2-State Data Types	26
2.2 Fixed-Size Arrays	27
2.2.1 Declaring and Initializing Fixed-Size Arrays	28
2.2.2 The Array Literal	29
2.2.3 Basic Array Operations — <i>for</i> and <i>Foreach</i>	30
2.2.4 Basic Array Operations – Copy and Compare	32
2.2.5 Bit and Array Subscripts, Together at Last	32
2.2.6 Packed Arrays	33
2.2.7 Packed Array Examples	33
2.2.8 Choosing Between Packed and Unpacked Arrays	34
2.3 Dynamic Arrays	35
2.4 Queues	36
2.5 Associative Arrays	38
2.6 Array Methods	41
2.6.1 Array Reduction Methods	41
2.6.2 Array Locator Methods	42
2.6.3 Array Sorting and Ordering	44
2.6.4 Building a Scoreboard with Array Locator Methods	45
2.7 Choosing a Storage Type	46
2.7.1 Flexibility	46
2.7.2 Memory Usage	46
2.7.3 Speed	47
2.7.4 Data Access	47
2.7.5 Choosing the Best Data Structure	48
2.8 Creating New Types with <code>typedef</code>	48
2.9 Creating User-Defined Structures	50
2.9.1 <i>Creating a Struct and a New Type</i>	50
2.9.2 Initializing a Structure	51
2.9.3 Making a Union of Several Types	51
2.9.4 Packed Structures	52
2.9.5 Choosing Between Packed and Unpacked Structures	52
2.10 Packages	53
2.11 Type Conversion	54
2.11.1 The Static Cast	54
2.11.2 The Dynamic Cast	55
2.12 Streaming Operators	55
2.13 Enumerated Types	57
2.13.1 Defining Enumerated Values	58
2.13.2 Routines for Enumerated Types	59
2.13.3 Converting to and from Enumerated Types	60

2.14	Constants	61
2.15	Strings	61
2.16	Expression Width	62
2.17	Conclusion.....	63
2.18	Exercises	64
3	Procedural Statements and Routines	69
3.1	Procedural Statements	69
3.2	Tasks, Functions, and Void Functions	71
3.3	Task and Function Overview.....	72
3.3.1	Routine Begin...End Removed	72
3.4	Routine Arguments	72
3.4.1	C-style Routine Arguments.....	72
3.4.2	Argument Direction	73
3.4.3	Advanced Argument Types	73
3.4.4	Default Value for an Argument	75
3.4.5	Passing Arguments by Name	76
3.4.6	Common Coding Errors	77
3.5	Returning from a Routine.....	78
3.5.1	The Return Statement.....	78
3.5.2	Returning an Array from a Function.....	78
3.6	Local Data Storage.....	79
3.6.1	Automatic Storage.....	80
3.6.2	Variable Initialization	80
3.7	Time Values.....	81
3.7.1	Time Units and Precision	81
3.7.2	Time Literals	82
3.7.3	Time and Variables.....	82
3.7.4	\$time vs. \$realtime	83
3.8	Conclusion.....	83
3.9	Exercises	83
4	Connecting the Testbench and Design	87
4.1	Separating the Testbench and Design	88
4.1.1	Communication Between the Testbench and DUT.....	88
4.1.2	Communication with Ports.....	89
4.2	The Interface Construct.....	90
4.2.1	Using an Interface to Simplify Connections	91
4.2.2	Connecting Interfaces and Ports.....	93
4.2.3	Grouping Signals in an Interface Using Modports	94
4.2.4	Using Modports with a Bus Design	95
4.2.5	Creating an Interface Monitor	95
4.2.6	Interface Trade-Offs	96
4.2.7	More Information and Examples	97
4.2.8	Logic vs. Wire in an Interface	97

4.3	Stimulus Timing	98
4.3.1	Controlling Timing of Synchronous Signals with a Clocking Block.....	98
4.3.2	Timing Problems in Verilog	99
4.3.3	Testbench – Design Race Condition	100
4.3.4	The Program Block and Timing Regions.....	101
4.3.5	Specifying Delays Between the Design and Testbench ...	103
4.4	Interface Driving and Sampling	104
4.4.1	Interface Synchronization	104
4.4.2	Interface Signal Sample	105
4.4.3	Interface Signal Drive	106
4.4.4	Driving Interface Signals Through a Clocking Block.....	106
4.4.5	Bidirectional Signals in the Interface	108
4.4.6	Specifying Delays in Clocking Blocks	109
4.5	Program Block Considerations	110
4.5.1	The End of Simulation	110
4.5.2	Why are Always Blocks not Allowed in a Program?	111
4.5.3	The Clock Generator	111
4.6	Connecting It All Together.....	112
4.6.1	An Interface in a Port List Must Be Connected	113
4.7	Top-Level Scope.....	114
4.8	Program–Module Interactions.....	115
4.9	SystemVerilog Assertions	116
4.9.1	Immediate Assertions.....	116
4.9.2	Customizing the Assertion Actions.....	117
4.9.3	Concurrent Assertions	118
4.9.4	Exploring Assertions.....	118
4.10	The Four-Port ATM Router.....	119
4.10.1	ATM Router with Ports	119
4.10.2	ATM Top-Level Module with Ports	120
4.10.3	Using Interfaces to Simplify Connections	123
4.10.4	ATM Interfaces.....	124
4.10.5	ATM Router Model Using an Interface	124
4.10.6	ATM Top Level Module with Interfaces	125
4.10.7	ATM Testbench with Interface.....	125
4.11	The Ref Port Direction.....	126
4.12	Conclusion.....	127
4.13	Exercises	128
5	Basic OOP.....	131
5.1	Introduction	131
5.2	Think of Nouns, not Verbs	132
5.3	Your First Class	133
5.4	Where to Define a Class	133
5.5	OOP Terminology	134

5.6	Creating New Objects	135
5.6.1	Handles and Constructing Objects	135
5.6.2	Custom Constructor	136
5.6.3	Separating the Declaration and Construction.....	137
5.6.4	The Difference Between New() and New[].....	138
5.6.5	Getting a Handle on Objects	138
5.7	Object Deallocation.....	139
5.8	Using Objects.....	140
5.9	Class Methods	141
5.10	Defining Methods Outside of the Class	142
5.11	Static Variables vs. Global Variables	143
5.11.1	A Simple Static Variable.....	143
5.11.2	Accessing Static Variables Through the Class Name	144
5.11.3	Initializing Static Variables	145
5.11.4	Static Methods.....	145
5.12	Scoping Rules.....	146
5.12.1	What is This?.....	148
5.13	Using One Class Inside Another	149
5.13.1	How Big or Small Should My Class Be?.....	151
5.13.2	Compilation Order Issue	151
5.14	Understanding Dynamic Objects	152
5.14.1	Passing Objects and Handles to Methods	152
5.14.2	Modifying a Handle in a Task.....	153
5.14.3	Modifying Objects in Flight.....	154
5.14.4	Arrays of Handles	155
5.15	Copying Objects.....	156
5.15.1	Copying an Object with the <i>New</i> Operator.....	156
5.15.2	Writing Your Own Simple Copy Function.....	158
5.15.3	Writing a Deep Copy Function	159
5.15.4	Packing Objects to and from Arrays Using Streaming Operators.....	161
5.16	Public vs. Local	162
5.17	Straying Off Course	163
5.18	Building a Testbench.....	163
5.19	Conclusion.....	164
5.20	Exercises	165
6	Randomization	169
6.1	Introduction	169
6.2	What to Randomize.....	170
6.2.1	Device Configuration	170
6.2.2	Environment Configuration.....	171
6.2.3	Primary Input Data.....	171
6.2.4	Encapsulated Input Data	171
6.2.5	Protocol Exceptions, Errors, and Violations	172
6.2.6	Delays.....	172

6.3	Randomization in SystemVerilog.....	172
6.3.1	Simple Class with Random Variables	173
6.3.2	Checking the Result from Randomization	174
6.3.3	The Constraint Solver	175
6.3.4	What can be Randomized?.....	175
6.4	Constraint Details.....	175
6.4.1	Constraint Introduction	176
6.4.2	Simple Expressions	176
6.4.3	Equivalence Expressions.....	177
6.4.4	Weighted Distributions.....	177
6.4.5	Set Membership and the Inside Operator.....	179
6.4.6	Using an Array in a Set	180
6.4.7	Bidirectional Constraints.....	183
6.4.8	Implication Constraints	184
6.4.9	Equivalence Operator.....	186
6.5	Solution Probabilities	186
6.5.1	Unconstrained	187
6.5.2	Implication	187
6.5.3	Implication and Bidirectional Constraints	188
6.5.4	Guiding Distribution with <code>Solve...Before</code>	189
6.6	Controlling Multiple Constraint Blocks.....	191
6.7	Valid Constraints	192
6.8	In-Line Constraints.....	192
6.9	The <code>pre_randomize</code> and <code>post_randomize</code> Functions.....	193
6.9.1	Building a Bathtub Distribution	193
6.9.2	Note on Void Functions.....	195
6.10	Random Number Functions	195
6.11	Constraints Tips and Techniques.....	196
6.11.1	Constraints with Variables.....	196
6.11.2	Using Nonrandom Values	197
6.11.3	Checking Values Using Constraints	198
6.11.4	Randomizing Individual Variables	198
6.11.5	Turn Constraints Off and On.....	198
6.11.6	Specifying a Constraint in a Test Using In-Line Constraints.....	199
6.11.7	Specifying a Constraint in a Test with External Constraints.....	199
6.11.8	Extending a Class	200
6.12	Common Randomization Problems	200
6.12.1	Use Signed Variables with Care.....	201
6.12.2	Solver Performance Tips	202
6.12.3	Choose the Right Arithmetic Operator to Boost Efficiency	202
6.13	Iterative and Array Constraints	203
6.13.1	Array Size.....	203
6.13.2	Sum of Elements	203

6.13.3	Issues with Array Constraints	205
6.13.4	Constraining Individual Array and Queue Elements	207
6.13.5	Generating an Array of Unique Values	208
6.13.6	Randomizing an Array of Handles.....	211
6.14	Atomic Stimulus Generation vs. Scenario Generation	211
6.14.1	An Atomic Generator with History.....	212
6.14.2	Random Array of Objects	212
6.14.3	Combining Sequences.....	213
6.14.4	Randsequence.....	213
6.15	Random Control.....	215
6.15.1	Introduction to <code>randcase</code>	215
6.15.2	Building a Decision Tree with <code>randcase</code>	216
6.16	Random Number Generators.....	217
6.16.1	Pseudorandom Number Generators	217
6.16.2	Random Stability — Multiple Generators	217
6.16.3	Random Stability and Hierarchical Seeding	219
6.17	Random Device Configuration.....	220
6.18	Conclusion.....	223
6.19	Exercises	224
7	Threads and Interprocess Communication	229
7.1	Working with Threads.....	230
7.1.1	Using <code>fork...join</code> and <code>begin...end</code>	231
7.1.2	Spawning Threads with <code>fork...join_none</code>	232
7.1.3	Synchronizing Threads with <code>fork...join_any</code>	233
7.1.4	Creating Threads in a Class.....	234
7.1.5	Dynamic Threads	235
7.1.6	Automatic Variables in Threads	236
7.1.7	Waiting for all Spawned Threads	238
7.1.8	Sharing Variables Across Threads	239
7.2	Disabling Threads	240
7.2.1	Disabling a Single Thread.....	241
7.2.2	Disabling Multiple Threads.....	241
7.2.3	Disable a Task that was Called Multiple Times	243
7.3	Interprocess Communication	244
7.4	Events.....	244
7.4.1	Blocking on the Edge of an Event.....	245
7.4.2	Waiting for an Event Trigger.....	245
7.4.3	Using Events in a Loop	246
7.4.4	Passing Events.....	247
7.4.5	Waiting for Multiple Events.....	248
7.5	Semaphores	250
7.5.1	Semaphore Operations	251
7.5.2	Semaphores with Multiple Keys	252

7.6	Mailboxes.....	252
7.6.1	Mailbox in a Testbench.....	255
7.6.2	Bounded Mailboxes	256
7.6.3	Unsynchronized Threads Communicating with a Mailbox.....	257
7.6.4	Synchronized Threads Using a Bounded Mailbox and a Peek	259
7.6.5	Synchronized Threads Using a Mailbox and Event.....	261
7.6.6	Synchronized Threads Using Two Mailboxes	262
7.6.7	Other Synchronization Techniques.....	264
7.7	Building a Testbench with Threads and IPC	264
7.7.1	Basic Transactor	265
7.7.2	Configuration Class	266
7.7.3	Environment Class.....	266
7.7.4	Test Program.....	267
7.8	Conclusion	268
7.9	Exercises	269
8	Advanced OOP and Testbench Guidelines.....	273
8.1	Introduction to Inheritance	274
8.1.1	Basic Transaction.....	275
8.1.2	Extending the <code>Transaction</code> Class.....	275
8.1.3	More OOP Terminology	277
8.1.4	Constructors in Extended Classes.....	277
8.1.5	Driver Class	278
8.1.6	Simple Generator Class	279
8.2	Blueprint Pattern	280
8.2.1	The <code>Environment</code> Class.....	281
8.2.2	A Simple Testbench.....	282
8.2.3	Using the Extended <code>Transaction</code> Class	283
8.2.4	Changing Random Constraints with an Extended Class.....	283
8.3	Downcasting and Virtual Methods	284
8.3.1	Downcasting with <code>\$cast</code>	284
8.3.2	Virtual Methods	286
8.3.3	Signatures and Polymorphism	288
8.3.4	Constructors are Never Virtual	288
8.4	Composition, Inheritance, and Alternatives.....	288
8.4.1	Deciding Between Composition and Inheritance	288
8.4.2	Problems with Composition	289
8.4.3	Problems with Inheritance	291
8.4.4	A Real-World Alternative	292
8.5	Copying an Object	293
8.5.1	Specifying a Destination for Copy	294
8.6	Abstract Classes and Pure Virtual Methods.....	295

8.7	Callbacks	297
8.7.1	Creating a Callback	298
8.7.2	Using a Callback to Inject Disturbances	299
8.7.3	A Quick Introduction to Scoreboards	300
8.7.4	Connecting to the Scoreboard with a Callback	300
8.7.5	Using a Callback to Debug a Transactor	302
8.8	Parameterized Classes	302
8.8.1	A Simple Stack	302
8.8.2	Sharing Parameterized Classes	305
8.8.3	Parameterized Class Suggestions	305
8.9	Static and Singleton Classes	306
8.9.1	Dynamic Class to Print Messages	306
8.9.2	Singleton Class to Print Messages	307
8.9.3	Configuration Database with Static Parameterized Class	308
8.10	Creating a Test Registry	311
8.10.1	Test registry with Static Methods	311
8.10.2	Test Registry with a Proxy Class	313
8.10.3	UVM Factory Build	319
8.11	Conclusion	319
8.12	Exercises	320
9	Functional Coverage	323
9.1	Gathering Coverage Data	324
9.2	Coverage Types	326
9.2.1	Code Coverage	326
9.2.2	Functional Coverage	327
9.2.3	Bug Rate	327
9.2.4	Assertion Coverage	328
9.3	Functional Coverage Strategies	328
9.3.1	Gather Information, not Data	328
9.3.2	Only Measure What you are Going to Use	329
9.3.3	Measuring Completeness	329
9.4	Simple Functional Coverage Example	330
9.5	Anatomy of a Cover Group	333
9.5.1	Defining a Cover Group in a Class	334
9.6	Triggering a Cover Group	335
9.6.1	Sampling Using a Callback	335
9.6.2	Cover Group with a User Defined Sample Argument List	336
9.6.3	Cover Group with an Event Trigger	337
9.6.4	Triggering on a System Verilog Assertion	337
9.7	Data Sampling	338
9.7.1	Individual Bins and Total Coverage	338
9.7.2	Creating Bins Automatically	339

9.7.3	Limiting the Number of Automatic Bins Created.....	339
9.7.4	Sampling Expressions	340
9.7.5	User-Defined Bins Find a Bug	341
9.7.6	Naming the Cover Point Bins.....	342
9.7.7	Conditional Coverage.....	344
9.7.8	Creating Bins for Enumerated Types	345
9.7.9	Transition Coverage	345
9.7.10	Wildcard States and Transitions.....	346
9.7.11	Ignoring Values	346
9.7.12	Illegal Bins	347
9.7.13	State Machine Coverage.....	347
9.8	Cross Coverage	348
9.8.1	Basic Cross Coverage Example	348
9.8.2	Labeling Cross Coverage Bins.....	349
9.8.3	Excluding Cross Coverage Bins.....	351
9.8.4	Excluding Cover Points from the Total Coverage Metric	351
9.8.5	Merging Data from Multiple Domains	352
9.8.6	Cross Coverage Alternatives	352
9.9	Generic Cover Groups.....	354
9.9.1	Pass Cover Group Arguments by Value	354
9.9.2	Pass Cover Group Arguments by Reference.....	355
9.10	Coverage Options.....	355
9.10.1	Per-Instance Coverage.....	355
9.10.2	Cover Group Comment	356
9.10.3	Coverage Threshold	357
9.10.4	Printing the Empty Bins	357
9.10.5	Coverage Goal.....	357
9.11	Analyzing Coverage Data	358
9.12	Measuring Coverage Statistics During Simulation	359
9.13	Conclusion.....	360
9.14	Exercises	360
10	Advanced Interfaces	363
10.1	Virtual Interfaces with the ATM Router.....	364
10.1.1	The Testbench with Just Physical Interfaces.....	364
10.1.2	Testbench with Virtual Interfaces.....	366
10.1.3	Connecting the Testbench to an Interface in Port List.....	369
10.1.4	Connecting the Test to an Interface with an XMR.....	370
10.2	Connecting to Multiple Design Configurations	372
10.2.1	A Mesh Design.....	372
10.2.2	Using <code>Typedefs</code> with Virtual Interfaces	375
10.2.3	Passing Virtual Interface Array Using a Port.....	376
10.3	Parameterized Interfaces and Virtual Interfaces.....	377

10.4	Procedural Code in an Interface	379
10.4.1	Interface with Parallel Protocol	380
10.4.2	Interface with Serial Protocol	380
10.4.3	Limitations of Interface Code	381
10.5	Conclusion	382
10.6	Exercises	382
11	A Complete SystemVerilog Testbench	385
11.1	Design Blocks	385
11.2	Testbench Blocks	390
11.3	Alternate Tests	411
11.3.1	Your First Test - Just One Cell	411
11.3.2	Randomly Drop Cells	412
11.4	Conclusion	413
11.5	Exercises	414
12	Interfacing with C/C++	415
12.1	Passing Simple Values	416
12.1.1	Passing Integer and Real Values	416
12.1.2	The Import Declaration	416
12.1.3	Argument Directions	417
12.1.4	Argument Types	418
12.1.5	Importing a Math Library Routine	419
12.2	Connecting to a Simple C Routine	419
12.2.1	A Counter with Static Storage	420
12.2.2	The Chandle Data Type	421
12.2.3	Representation of Packed Values	423
12.2.4	4-State Values	424
12.2.5	Converting from 2-State to 4-State	426
12.3	Connecting to C++	427
12.3.1	The Counter in C++	427
12.3.2	Static Methods	428
12.3.3	Communicating with a Transaction Level C++ Model	428
12.4	Simple Array Sharing	432
12.4.1	Single Dimension Arrays - 2-State	432
12.4.2	Single Dimension Arrays - 4-State	433
12.5	Open arrays	434
12.5.1	Basic Open Array	434
12.5.2	Open Array Methods	435
12.5.3	Passing Unsized Open Arrays	435
12.5.4	Packed Open Arrays in DPI	436
12.6	Sharing Composite Types	437
12.6.1	Passing Structures Between SystemVerilog and C	438
12.6.2	Passing Strings Between SystemVerilog and C	439

12.7	Pure and Context Imported Methods	440
12.8	Communicating from C to SystemVerilog	441
12.8.1	A simple Exported Function	441
12.8.2	C function Calling SystemVerilog Function	442
12.8.3	C Task Calling SystemVerilog Task	444
12.8.4	Calling Methods in Objects	446
12.8.5	The Meaning of Context	449
12.8.6	Setting the Scope for an Imported Routine	450
12.9	Connecting Other Languages	452
12.10	Conclusion	453
12.11	Exercises	453
References		455
Index		457

SystemVerilog for Verification

A Guide to Learning the Testbench Language Features

Spear, C.; Tumbush, G.

2012, XLIV, 464 p., Hardcover

ISBN: 978-1-4614-0714-0