

# Contents

---

<b>Preface</b> .....	<b>xv</b>
<b>1 Introduction to Metric-Driven Verification</b> .....	<b>1</b>
1.1 Introduction .....	1
1.2 Failing to Plan = Planning to Fail .....	2
1.3 Metric-Driven Verification .....	4
1.4 Building Strong Testbench Foundations .....	5
1.5 Simulation Isn't the Only Way .....	7
1.6 Low Power isn't Just the Designer's Problem .....	8
1.7 Reuse Isn't Just About Testbench Components .....	9
1.8 Does Speed Matter? .....	10
1.9 What About Scalability? .....	11
1.10 Is Metric-Driven Verification Just for RTL Hardware? .....	12
1.11 How Do I Get Up to Speed with All this New Stuff? .....	14
1.12 Summary .....	14
<b>2 UVM and Metric-Driven Verification for Mixed-Signal</b> .....	<b>15</b>
2.1 Why Metric-Driven Verification for Analog? .....	15
2.2 Approach and Scope .....	16
2.3 Planning for Analog Verification .....	20
2.3.1 Including Analog Properties .....	20
2.3.2 Verification Plan Structured for Reuse .....	22
2.4 Constructing a UVM-MS Verification Environment .....	23
2.4.1 Analog Verification Blocks .....	23
2.4.2 Analog Configuration Interface .....	24
2.4.3 Threshold Crossing Monitor .....	24
2.5 Architecture of a Sample Testbench .....	25

2.6	Collecting Coverage .....	26
2.6.1	Direct and Computed Coverage Collection .....	27
2.6.2	Deciding on Coverage Ranges .....	30
2.6.3	Trading Off Speed and Visibility .....	30
2.7	Generating Inputs .....	32
2.7.1	Dealing with Configurations and Settings .....	32
2.7.2	Generating and Driving Digital Control .....	33
2.8	Checking Analog Functionality .....	38
2.8.1	Comparing Two Values .....	38
2.8.2	Triggering a Check on Control Changes .....	40
2.8.3	Measuring Signal Timing .....	43
2.8.4	Comparing a Value to a Threshold .....	45
2.8.5	Checking Frequency Response .....	47
2.9	Using Assertions .....	49
2.9.1	Checking Input Conditions .....	49
2.9.2	Verifying Local Invariants .....	50
2.9.3	Limitations on Assertion Checking of Analog Properties .....	50
2.9.4	Dealing with Different Modeling Styles .....	50
2.10	Clocks, Resets and Power Controls .....	51
2.10.1	Driving Clocks .....	51
2.10.2	Resets .....	51
2.10.3	Power-Up and Power-Down Sequences .....	52
2.11	Analog Model Creation and Validation .....	54
2.12	Integrating the Test Environment .....	54
2.12.1	Connecting the Testbench .....	55
2.12.2	Connecting to Electrical Nodes .....	55
2.12.3	System-Level Parameters and Timing .....	56
2.12.4	Supporting Several Model Styles In A Single Testbench .....	57
2.12.5	Interfacing Between Real and Electrical Signals .....	60
2.12.6	Creating Run Scripts and Other Support Files .....	61
2.12.7	Recommended Directory Structure .....	62
2.13	Closing the Loop Between Regressions and Plan .....	63
2.13.1	Implementation of Coverage for Analog .....	63
2.13.2	Updating the Verification Plan With Implementation Data .....	65
2.14	Regression Runs for Analog IP .....	66
2.14.1	Single Simulation Runs .....	66
2.14.2	Regressions—Running Multiple Test Cases .....	68
2.15	Moving Up to the SoC Level .....	70
2.15.1	Mix-and-Match SoC-Level Simulation .....	70
2.15.2	Updating the SoC-Level Test Plan .....	71

2.15.3	Integrating Into the SoC-Level Testbench	71
2.16	UVM-MS Universal Verification Blocks	73
2.16.1	Wire Verification Component	73
2.16.2	Simple register UVC	79
2.16.3	Analog to Digital Converter (ADC) UVC	83
2.16.4	Digital to Analog Converter (DAC) UVC	85
2.16.5	Level Crossing Monitor	87
2.16.6	Ramp Generator and Monitor	90
2.17	Summary	95
<b>3</b>	<b>Low-Power Verification with the UVM</b>	<b>97</b>
3.1	Introduction	97
3.1.1	What is Unique about Low-Power Verification	98
3.1.2	Understanding the Scope of Low-Power Verification	99
3.1.3	Low-Power Verification Methodology	100
3.1.4	Understanding Low-Power Verification	102
3.2	Understanding Low-Power Design and Verification Challenges	102
3.2.1	How Low-Power Implementations Are Designed Today	102
3.2.2	Challenges for Low-Power Verification	103
3.2.3	Low-Power Optimization	104
3.2.4	Low-Power Architectures	105
3.2.5	Low-Power Resources	111
3.3	Low-Power Verification Methodology	111
3.4	Low-Power Discovery and Verification Planning	113
3.4.1	Low-Power Discovery	113
3.4.2	Verification Planning	113
3.4.3	System-Level Planning	113
3.4.4	Hierarchical Planning	117
3.4.5	Domain-Level Verification Planning	118
3.4.6	A Note on Verifying Low-Power Structures	119
3.4.7	Recommendations for Designs with a Large Number of Power Modes	119
3.5	Creating a Power-Aware UVM Environment	121
3.5.1	Tasks for a Low-Power Verification Environment	121
3.5.2	Solution: Low-Power UVC	122
3.5.3	UVC Monitor	124
3.5.4	LP Sequence Driver	126
3.5.5	UVM-Based Power-Aware Verification	128
3.6	Executing the Low-Power Verification Environment	128
3.6.1	LP Design and Equivalency Checking	128
3.6.2	Low-Power Structural and Functional Checks	129

3.6.3	Requirements for Selecting a Simulator and Emulator .....	130
3.6.4	Advanced Debug and Visualizations .....	132
3.6.5	Automated Assertions and Coverage .....	135
3.6.6	Legal Power Modes and Transitions .....	135
3.6.7	Automatic Checking of Power Control Sequences .....	135
3.6.8	Verification Plan Generated from Power Intent .....	136
3.7	Common Low-Power Issues .....	138
3.7.1	Power-Control Issues .....	138
3.7.2	Domain Interfaces .....	139
3.7.3	System-Level Control .....	140
3.8	Summary .....	141

**4 Multi-Language UVM ..... 143**

4.1	Overview of UVM Multi-Language .....	143
4.2	UVC Requirements .....	146
4.2.1	Providing an Appropriate Configuration .....	146
4.2.2	Exporting Collected Information to Higher Levels .....	146
4.2.3	Providing Support for Driving Sequences from Other Languages .....	146
4.2.4	Providing the Foundation for Debugging of All Components .....	146
4.2.5	Optional Interfaces and Capabilities .....	147
4.3	Fundamentals of Connecting <i>e</i> and SystemVerilog .....	147
4.3.1	Type Conversion .....	147
4.3.2	Function Calls Across Languages .....	150
4.3.3	Passing Events Across Languages .....	153
4.4	Configuring Messaging .....	154
4.5	<i>e</i> Over Class-Based SystemVerilog .....	154
4.5.1	Environment Architecture .....	155
4.5.2	Configuration .....	156
4.5.3	Generating and Injecting Stimuli .....	160
4.5.4	Monitoring and Checking .....	168
4.6	SystemVerilog Class-Based over <i>e</i> .....	171
4.6.1	Simulation Flow in Mixed <i>e</i> and SystemVerilog Environments .....	173
4.6.2	Contacting Cadence for Further Information .....	173
4.7	UVM SystemC Methodology in Multi-Language Environments .....	174
4.7.1	Introduction to UVM SystemC .....	174
4.7.2	Using the Library Features for Modeling and Verification .....	175
4.7.3	Connecting between Languages using TLM Ports .....	182
4.7.4	Example of SC Reference Model used in SV Verification Environment .....	188
4.7.5	Reusing SystemC Verification Components .....	191
4.8	Summary .....	193

---

<b>5</b>	<b>Developing Acceleratable Universal Verification Components (UVCs)</b>	<b>. 195</b>
5.1	Introduction to UVM Acceleration	195
5.2	UVC Architecture	196
5.2.1	Standard UVC Architecture	196
5.2.2	Active Agent	196
5.2.3	Passive Agent	197
5.2.4	Acceleratable UVCs	197
5.3	UVM Acceleration Package Interfaces	201
5.3.1	uvm_accel_pipe_proxy_base Task and Function Definitions (SystemVerilog)	202
5.3.2	uvm_accel_pipe_proxy_base Task and Function Definitions ( <i>e</i> )	204
5.4	SCE-MI Hardware Interface	205
5.4.1	SCE-MI Input Pipe Interface	206
5.4.2	SCE-MI Output Pipe Interface	206
5.5	Building Acceleratable UVCs in SystemVerilog	207
5.5.1	Data Items	207
5.5.2	Acceleratable Driver (SystemVerilog)	209
5.6	Building Acceleratable UVCs in <i>e</i>	<b>215</b>
5.6.1	Data Items	215
5.6.2	Acceleratable Driver ( <i>e</i> )	216
5.7	Collector and Monitor	219
5.8	Summary	219
<b>6</b>	<b>Summary</b>	<b>221</b>
	<b>The Authors</b>	<b>223</b>
	<b>Index</b>	<b>227</b>

