# *Sunburst Design - Advanced Universal Verification Methodology*

by Recognized Verilog & SystemVerilog Guru, Cliff Cummings of Sunburst Design, Inc.

**3 Days**
**70% Lecture, 30% Lab**

**Prerequisites (mandatory) -** ***This is an advanced UVM verification course that assumes engineers have already taken UVM training or have 2 years of UVM experience.***

## Course Syllabus

***(~10 minute breaks near the top of each hour)***
***(Lab time is scheduled for "Lunch & Lab" and again near the end of the day)***

*This course may be customized by client companies on a WebEx conference call with Cliff Cummings.*

## Day One

### UVM Resources & Introduction

- UVM resources
- Industry trends - UVM verification
- Industry trends - debug time related to project schedule

### Review of Advanced Techniques Used in UVM Base Classes

- Up-casting and down-casting (used extensively in UVM verification environments)
- Local & protected (hiding) in UVM Base Class Library (BCL)
- Static class methods in UVM BCL
- Extern methods in UVM BCL
- Singleton pattern and usage in UVM BCL

### Review of Best UVM Reporting Macro Techniques

*Includes materials from Cliff's SNUG 2014 award-winning paper on UVM messaging.*

- UVM messaging messages & macros - emphasis on macros
- UVM verbosities - why you should avoid using UVM_LOW verbosity
- UVM verbosity usage guidelines

## File Guards, Packages & Command File Strategies for Large Projects

- Importing from packages - multiple techniques
- SystemVerilog-2009 - importing packages in module headers
- SystemVerilog-2009 - importing and exporting nested packages
- File guards & recommended naming convention
- File guards for macro files - compile in command files first
- Incdir command options - compile in command files second
- Nested command files - compile in command files third
- File guards in interface files
- File guards in package files
- Each class a separate file
- Extern methods in class files
- Include class files into package files
- Keep packages out of the global space
- Compiling packages

## Advanced uvm_resource_db Techniques

*Includes materials from Cliff's DVCon 2023 paper on UVM Resources the uvm_resource_db API.*

- Comparing the OVM set_config_* commands, uvm_config_db API and uvm_resource_db API
- Deep dive into how the set_config_* commands work and their disadvantages
- Why engineers should not use assert on uvm_config_db#()::get commands
- Why the OVM set_config_* commands were deprecated from UVM
- Deep dive into the UVM resources database
- Why 95%+ of engineers use uvm_config_db and why they should use uvm_resource_db
- The uvm_config_db API and why it has limitations
- The uvm_resource_db and how it removes the uvm_config_db limitations
- Favorable experiences using uvm_resource_db on a recent, huge verification project
- LAB: uvm_config_db and uvm_resource_db usage *(Full UVM self-checking testbench)*

## Advanced Virtual Interfaces Techniques I

*Includes materials from Cliff's SNUG 2021 award-winning paper on VIF-Harness Techniques.*

- Common virtual interface styles
- Style #1 - DUT interface hierarchical connections
- Style #2 - DUT interface port connections

## Advanced Virtual Interfaces Techniques II

*Includes materials from Cliff's SNUG 2021 award-winning paper on VIF-Harness Techniques.*

- Harness / Bind virtual interface styles
- Style #3 - Larson-Harness bind technique
- Style #4 - Bind-Harness-dut_if connections
- Style #5 - Bind-dut_if connections - New technique shown at SNUG 2021
- LAB: Bind-dut_if connections testbench *(Full UVM self-checking testbench)*

### UVM Testbench Environment with Config Objects

- Config objects store configuration information in components
- Config objects extend from uvm_object
- Most common usage: tests, environments, agents
- Active and passive agents
- Enabling functional coverage component
- Passing configuration information from test to environment to agent
- Multi-part config object example
- LAB – FIFO Gray Code Pointer - *(Full UVM self-checking testbench)*

### SystemVerilog Bind Command

- Bindfile input ports
- Bindfile connected using .* port connections
- Bind command placed in the testbench
- How the bind command works
- Bindfile connected using combination of .* and named ports
- Bindfile signal declarations?
- Bind command placed in a second top-level dummy module

## Day Two

### Review of UVM Transaction Definition Types & Sequence Definition Types

*Includes materials from Cliff's SNUG 2014 award-winning paper on UVM transactions.*

- Why classes -vs- structs?
- do_copy, do_compare and other do_methods
- Field macro limitations
- UVM sequence body task
- pre_start() -vs- pre_body()
- start_item(tx) - finish_item(tx)
- `uvm_do macros
- Benchmarks

### Review of UVM Scoreboard Style #1

*Includes materials from Cliff's SNUG 2013 paper on UVM scoreboard architectures.*

- SystemVerilog queues
- SystemVerilog mailboxes
- uvm_tlm_fifo
- uvm_tlm_analysis_fifo
- Scoreboard architecture style #1
- Pre-coded scoreboard wrapper and predictor
- Extern calc_exp function - requires user to complete this function
- Pre-coded comparator with 2 uvm_tlm_analysis_fifos
- LAB – UVM Scoreboard Style #1 - Barrel Shifter - *(Full UVM testbench lab)*
- LAB – UVM Scoreboard Style #1 - Pipeline Design - *(Full UVM testbench lab)*

### Review of Multiple Analysis Implementation Port Techniques

*Includes more materials from Cliff's SNUG 2013 paper on UVM scoreboard architectures.*

- Scoreboard architecture style #2
- Multiple analysis implementation ports
- `uvm_analysis_imp_decl macros
- LAB – UVM Scoreboard Style #2 - 2 Analysis Imp Ports - *(Full UVM testbench lab)*

### Reactive Stimulus Techniques Using the Agent-Sequencer

*Includes materials from Cliff's DVCon 2020 award-winning paper on Reactive Stimulus.*

- Explanation of the reactive driver
- Explanation of the reactive sequencer
- Explanation of the reactive sequence
- Sampling output fields into the response transaction
- Common response coding mistake
- VIP considerations
- LAB: Sequencer-driver reactive stimulus testbench *(Full UVM self-checking testbench)*

### Advanced Virtual Sequence Techniques

*Includes materials from Cliff's DVCon 2023 paper on UVM Resources the uvm_resource_db API.*

- Three virtual sequencer techniques are shown - advantages / disadvantages described
- Virtual sequence that retrieves subsequencer handles stored in a virtual sequencer
- Test_base with init_vseq() method to store subsequencer handles in the vseq_base
- Using the uvm_resource_db to retrieve subsequencer handles directly
- LABS: Equivalent virtual sequence labs using (1) Virtual sequencer, (2) init_vseq() method, (3) Using the uvm_resource_db *(Three Full UVM self-checking testbenches)*

## Day Three

### Review of Clocking Blocks & Verification Timing

*Includes materials from Cliff's SNUG 2016 paper on UVM verification timing techniques.*

- Testbench stimulus/verification vector timing strategies
- #1step sampling
- Clocking blocks
- Clocking skews
- UVM usage of clocking blocks in an interface
- UVM driver timing using clocking blocks
- UVM signal sampling using clocking blocks
- 3 important timing techniques (#1 - applying stimulus, #2 & #3 sampling for verification)
- LABS - All of the full UVM self-checking labs use these clocking block techniques

### Review of UVM Factory Overrides

*Includes materials from Cliff's SNUG 2012 paper on the UVM factory and overrides.*

- Introduction to factory overrides
- Review of factory override by_type
- Review of factory override by_inst

### UVM Parameterized DUT Interface - Fundamental Technique

- Passing top-module & DUT parameters to the UVM testbench
- `uvm_component_param_utils
- `uvm_object_para_utils
- Testbench components modified for parameterized testing
- Testbench transaction/sequences modified for parameterized testing
- +UVM_TESTNAME & factory modifications / setup
- Why this technique is tedious
- LAB: UVM parameterized DUT interface *(Full UVM self-checking testbench)*

### UVM Parameterized DUT Interface - Advanced dut_max_if Technique

- DUT Max Interface Technique - simplifies testing of parameterized designs
- dut_max_if / dut_if - connecting different bus sizes
- Bind dut_if inside of DUT
- max_defines.sv file
- DUT information struct: dut_info_s
- Port coercion
- Trick to set proper printing widths
- LAB: UVM parameterized DUT-Max interface *(Full UVM self-checking testbench)*

### Advanced Reactive Stimulus Techniques Using the Monitor and uvm_tlm_analysis_fifo

*Includes materials from Cliff's DVCon 2021 award-winning paper on Advanced Reactive Stimulus.*

- Sampling the stimulus response from the same agent
- Sampling the stimulus response from a 2nd agent
- Two additional reactive stimulus techniques
- uvm_tlm_analysis_fifo in the environment
- Base_sequence using blocking-get to retrieve output sequence from uvm_tlm_analysis_fifo
- Base_sequence triggers a response event
- Using pre_start() -vs- pre_body() method
- Second technique using config objects and a virtual sequencer
- LAB: Multi-agent reactive stimulus testbench *(Full UVM self-checking testbench)*

### Additional Advanced Techniques

- SystemVerilog-2012 interface classes
- DUT error injection without recompiling the DUT
- DUT error injection using bindfile-force technique
- Multi-agent packet example
- LAB: DUT error injection using bindfile *(Full UVM self-checking testbench)*
- LAB: Multi-agent packet example *(Full UVM self-checking testbench)*