Boosting Debug Productivity
Practical Applications of Debug Innovations in a UVM World

Joerg Richter
Director R&D, Verification Group
Agenda

Motivation

Efficient SV/UVM Testbench Debug
Rapid Root-Cause Analysis with Reverse Debug
Search-centric Debug with OneSearch
Embedded SW Debug
Verification Complexity Exponentially Increasing

- 10x
  - 10M+ gates ASIC
  - 5-10 IPs per ASIC
  - 10+ power domains
  - 10-15 clocks
  - 1-2 interface protocols
  - 100K lines of SW
- 10x
  - 100M+ gates mobile SoC
  - 50+ IPs per SoC
  - 100+ power domains
  - 300+ clocks
  - 30+ interface protocols
  - 10M+ lines of SW

- Billion+ Gates SoCs
- Functionality
- Power Performance
- Block-level testing to SW Bring-up
- Shrinking Time-to-Market Windows

Logic States & Transitions

Source: Synopsys

© Accellera Systems Initiative
Scope of Debug Expanding

- Constraints
- Coverage
- Software
- TestBench
- UPF
- Embedded Processor
- Analog
- Bus Protocols
- VIP

© Accellera Systems Initiative
Motivation

Efficient SV/UVM Testbench Debug
- Rapid Root-Cause Analysis with Reverse Debug
- Search-centric Debug with OneSearch

Embedded SW Debug
Testbench Debug Complexity

Verification Approach
- Migration from v2k to SystemVerilog/UVM
- Expertise mismatch

Environment
- Languages & Methodologies
- In-house/vendor VIP

Scale
- Block to sub-system to system
- Signal-level to Protocol-level

Volume
- Amount & complexity of functionality to verify
- Use of IPs in design
Debug Testbenches Interactively

Full-visibility Testbench Debug in Verdi & VCS

- On-demand Debug
- Step forward/back/into testbench constructs
- Go back in time without restarting simulation OR setting checkpoints

- Full Visibility
  - Source code, dynamic objects, watch list etc.
  - Call stack, members, local variables etc.

- Powerful Capabilities
  - Trace value assignment into testbench
  - Perform what-if analysis
  - Enable day-to-day and batch simulation debug
Key Elements for Efficient SystemVerilog TB Debug

- **Class Browser**: Class hierarchy, methods, members
- **Object Browser**: All objects with member values and UVM hierarchy
- **Class Browser**: Class instances with member values
- **Stack and Locals**: Call stack of thread(s) Frame locals with values

© Accellera Systems Initiative
Simulation Control & TB Behavior Tracking

SmartLog: Powerful Interactive Console

Simulation Control

Breakpoints

Source with Value Annotation

Watch
Making Debug UVM Aware

Resource DB

Factory

Domains / Phases

Sequences

Registers

Hierarchy

© Accellera Systems Initiative 10
Buried in logfiles?

- Logfile is too big! How do I find the UVM Fatal?
- Emacs search not enough anymore...
- Manually correlating log messages is so annoying!
- I want just the errors around the 10,000 nsec mark
- What’s the source code that created this message?
- How can I browse a log with custom messages?

There has to be a better way to browse data across all these logs!
SmartLog to the Rescue

- Smart browsing of log files
  - Hyperlinks
  - Message filtering: UVM, VCS, Severity...
  - Time visualization and navigation
  - Message navigation
  - File view or structure view
  - Searching and text based filtering
  - Customize and add own log parsing rules (for custom log files)
SmartLog – File View

File view / Structure view

Table view with columns
Agenda

Motivation
Efficient SV/UVM Testbench Debug
Rapid Root-Cause Analysis with Reverse Debug
Search-centric Debug with OneSearch
Embedded SW Debug
Productivity with Reverse Debug

Basic Interactive Debug

Simulation Time 0

Breakpoint

Stepped beyond line of Interest

Restart Simulation

Simulation Time 0

Rerun Simulation

Done

NO Restart Needed

Simulation Time 0

Go Back to ANY Point in Time

Step back

Saves Hours of Simulation

w/ Reverse Debug

© Accellera Systems Initiative
Finding Failures

Diagnose and Debug - Moving forward and backward in time

- Jump back to previous point in time
- Step forward through code to find issue

Test FAIL

Advance forward to later time without re-simulation

Simulation Time 0

Undo/Redo
- Last simulation control command
- Forward & Backward

Go To
- Previous/Next Value Assignment

Reverse Controls

© Accellera Systems Initiative
What-if Analysis
Fix and Validate in One Simulation

- Jump back to previous point in time
- Step forward through code to find issue
- Force signal – Expect new outcome
- Simulate forwards
- New outcome, test does not stop/fail here anymore
- Repeat if needed

Test FAIL

No need to recompile

Test PASS

No need to re-run simulation

Done

© Accellera Systems Initiative
Regression Debug

Overnight Regression

w/ Regression Debug Enabled

1. Analyze log; Identify events
2. Instrument checkpoints
3. Rerun sims interactively
4. Debug

Time Savings Per Test

Reduce debug TAT significantly
Maximize regression utilization
Focus manual effort on debug
Interactive Debug Flows

*Interactive debug for both day-to-day and batch simulation tasks*

1. User runs and debugs interactive simulation in Verdi

```
verdi
```

2. User runs batch/interactive simulation, saves sessions, later restores sessions and debugs interactively

```
simv
-ucli -ucli2Proc
```

© Accellera Systems Initiative 2017
Verdi Reverse Debug

LIVE

Demo
Agenda

Motivation
Efficient SV/UVM Testbench Debug
Rapid Root-Cause Analysis with Reverse Debug

Search-centric Debug with OneSearch

Embedded SW Debug
Search Queries...

Can you imagine life without Google?

What do you use Google for?

Any kind of Information! The only place!

What do you enter?

People just enter plain words – no syntax

How do you search?

Google Search  I'm Feeling Lucky

Do you still use bookmarks?

It changes your need of other features

Now, how would designers and verification engineers like to search?
OneSearch

A single search for the entire verification environment

• Single and simple user interface
• Fast search engines

Sources

searches within source files and files used for the design

Logs

searches within log files from compilation and execution

Identifiers

searches within the compiled design for any Identifier

Docs

searches within Verdi tool documentation PDF files

Your Own

searches fully customized
(custom DBs, specific files, environment, ...)

Custom PDFs: FAQ available

© Accellera Systems Initiative
Using Verdi OneSearch

Configure Settings
Search Query

Search Engine Selection
Hyperlinks to SmartLog Viewer
Hyperlinks to Source View
Hyperlinks to PDF Viewer

© Accellera Systems Initiative
Verdi OneSearch

Demo
Agenda

Motivation
Efficient SV/UVM Testbench Debug
Rapid Root-Cause Analysis with Reverse Debug
Search-centric Debug with OneSearch

Embedded SW Debug
SoC Debug Challenges
Lack of Visibility on the Software Side

Verdi
Design & Testbench Debug

CPU Software Debug
• Encrypted Model?
• Registers + Nets?
• No or Low-Visibility
• Looks like a Black Box

BUT... What about the embedded software?

Time Consuming Manual Debug:
• Log-files, Waveforms
• ELF Dump, Disassembly
• Decoding Stack, Variables

At every point in time...
SoC Debug Challenges

Lack of Visibility on the Software Side

Verdi HW SW Debug Add-on
Instruction-Accurate Embedded Processor Debug with Synchronized HDL, C, Assembly Visibility
Verdi HW SW Debug

- Enables co-debug between HW and SW
- HW and SW debug synchronized in time
- View C/Assembly source, C variables, stack, memory
- Debugs multiple cores simultaneously
- Simulation supports all ARM® Cortex® cores
- Easy to support additional cores or custom cores
Embedded Software Debug in Eclipse

Provides programmer’s view of code running on simulated processor

© Accellera Systems Initiative
Combined HW and SW Debug

Post Process Debug
- Eclipse feels like interactive debug
- Run Forward/Back
- Step Forward/Back
Debugging Multi-Core

Provides programmer’s views into multiple cores simultaneously
Verdi HW SW Debug Flow

Simulation
- SoC Design Files
- VCS simv
- CPU Log

Converter
- Verdi Pattern File
- hwsw_debug_convert

Debug
- Eclipse
- GDB
- HW/SW Engine
- Verdi
- Design FSDB
- HWSW FSDB
- C-Src & ELF files

C-Compile with ‘-g’ and optionally ‘–O1’
- C Src
- armcc -g
gcc -g
- ELF files
- HEX file

Existing RTL Flow
- New HW/SW Flow
- Existing C Flow

No modifications needed - Enable CPU log during simulation
Verdi HW SW Debug

Demo
Verdi HW SW: Getting Started

Install
- Download and install latest Verdi Release
- Run Dhrystone Example to verify installation

Prepare
- Identify the CPU cores in your design
- Dump CPU log from your simulation

Convert
- Run Converter on CPU log to generate FSDB

Debug
- Invoke Verdi and HW SW Debug/Eclipse
- Debug your Embedded Software and Design HDL
Summary

Interactive Debug
- Full-featured, full visibility debug for SV/UVM testbench
- Run forward and backwards for improved debug efficiency

OneSearch
- Fast and efficient search
- Search entire verification environment at once

SmartLog
- Organized logs in table format with time and other filters
- Parse logs with hyperlinks to source

HW SW Debug
- Synchronized HW and SW debug
- Multi-core support
- Performance profiling and embedded SW code coverage
Questions?

Motivation
Efficient SV/UVM Testbench Debug
Rapid Root-Cause Analysis with Reverse Debug
Search-centric Debug with OneSearch
Embedded SW Debug
References

- Synopsys Verification Home
  https://www.synopsys.com/verification.html

- Synopsys Verdi Debug Platform
  https://www.synopsys.com/verification/debug.html

- Synopsys Verdi HW/SW Debug
  https://www.synopsys.com/verification/debug/verdi-hw-sw-debug.html