UVM Debug using Visualizer Debug Environment

Intuitive, High Performance, High Capacity Debug Solution

Mark Handover
Where Verification Engineers are Spending Time

2 working days spent in debug every week

- Debugging (44%)
  - Complexity & Predictability

- Running Simulation (21%)
  - Speed & Capacity

- Testbench Development (19%)
  - Stimulus, Coverage, & Re-use

- Test Planning (14%)
  - Metrics, Analysis, & Process

Top Debug Challenges

*Productivity needs for today’s complex debug environment*

- **Capacity**
  - Capacity to load 500M+ gate designs in seconds

- **Debug efficiency**
  - Full debug visibility + fast simulation
  - Load large designs in seconds

- **Testbench debug**
  - Debug dynamic objects
  - Post and Interactive debug mode

- **Unified platform debugger**
  - Single debugger for Simulation, Emulation, Low Power, Formal
  - Reduced learning curve
Visualizer - Tackling Debug Complexity

*Single debug solution for complex heterogeneous verification environment*

- **All Engines**
  - Questa Simulation
  - Questa Formal Verification
  - Questa Clock Domain Checking
  - Questa Verification IP
  - Questa Low Power
  - Veloce Emulation
  - Veloce FPGA Prototyping
  - Calypto Synthesis

- **All Technologies**
  - UVM Debug
  - UPF Debug
  - RTL Design Debug
  - Transaction Level Debug
Visualizer Debug Environment

Features+ Performance + Memory Footprint + Capacity

- Simulation Performance
  - 2-4x simulation performance

- Memory
  - 2-3x reduction in memory footprint

- Capacity
  - Loads 500M+ gate designs in seconds
  - Go to debugger for large GLS designs

- UVM debug - live and Post sim
Full Design/TB Visibility by default
Reducing the latency of root-cause analysis

Classic debug flow

Regression ➔ Triage ➔ Run#2 ➔ Debug ➔ Run#3 ➔ Debug ➔ Run#4 ➔ Debug...

- Run fast! (no trace)
- Enable Debug Trace With 1st trace set
- Add more traces & data collection
- Still More Visibility Add $display() Debug

Visualizer debug flow

- Minimum trace set for fast simulation and compact results
- High performance “smart” reconstruction of signals
- Save Time by Capturing More Data, Faster, First Time
Debug UVM in Post Sim mode

- **Traditional debug**
  - I need interactive/live simulation to debug my UVM TB
  - Post Sim debug is for 1’s and 0’s

- **Visualizer debug**
  - Visualizer treats Class as a first class object
Simplified UVM Debug and Navigation

Intuitive Navigation of UVM TB

- **Traditional debug**
  - Manually trace connectivity to view and debug UVM TB

- **Visualizer**
  - Use UVM Schematic View as a Map
    - View existing TBs or use it as a navigation vehicle
  - View all handles and variables
  - Easy navigation by class hierarchy or schematic
Finding Config Database Mismatches

Simplified debug of Config objects

- The Config DB is a data exchange — Can set or get values anywhere in the testbench
Understand Hierarchy of Sequences

Intuitive Debug for Your UVM Sequences

- Traditional manual debug does not scale
- Visualizer Debug: Live sequence hierarchy activity by sequencer
UVM Class Debug in Waveform View

- Easy to drag into waveforms from source or object
- Add handles to waves, see member values over time
- Simple to see any Testbench transaction to your DUT
Live Simulation

- Interactively single step and add breakpoints
Visualizer – Enabling Cross-Platform Debug

Integrated UVM Questa and Veloce Debug

- Support TBX flow
  - Single GUI to view and debug waveform data from Simulator and emulator
**Visualizer™ Debug Environment**

- **Built for Speed**
  - High Performance, Capacity, Visibility
- **Increases Debug Flow Productivity**
  - Intuitive and easy-to-use
  - Powerful automation, find bugs fast
- **Supports Complex Projects**
  - Native SV Class Debug
  - UVM Navigation and Debug
- **Built for All Today’s Platforms**
  - Questa Simulation, Veloce Emulation
  - Formal, System-level, UPF