Post processing techniques to accelerate assertion development

Ajay Sharma
Agenda

Introduction to Assertions

Traditional flow for using ABV in Simulations/Emulation/Prototyping

Post processing Assertions using a structured and open events database.

Summary and Conclusions
Assertion Based Verification

Historical Perspective

• Assertions are not new!
  – Widely used in software development
    (Java // assert <expr> ; )
  – Designers frequently add verification code
    //synopsys translate_off
    if(full & new_req) $display ("Error: buffer overflow.");
    //synopsys translate_on
  – VHDL includes the keyword assert
  – The Accellera Open Verification Library
What is an Assertion?

• A piece of verification code used to check a property
  – correct/illegal behavior
  – assumptions/constraints
  – coverage goals

• Examples:
  – Interface A must follow the PCI protocol
  – Bus B must be one-hot
  – The FIFO must never overflow
  – I want to see back-to-back reads/writes
  – Write will follow read after 3 cycles
Why Use Assertions in Verification?

Traditional Flow Has Multiple Entry Points For Introducing Bugs

- **Text and timing diagrams are not tool consumable**
  - Separate manual conversion for every tool
  - Involves multiple persons, assumptions made
- **Need better way to communicate Spec to Verification tools**
Why Use Assertions in Verification?

• Concise Description of Temporal Behavior
  – Fewer lines of code, faster to write
  – Reduced chance of error
• Error Localization
  – White-box assertions may be placed close (physically and temporally) to possible bug sources
  – Faster debug of failures
• Block-to-System Portability
  – White-box and interface checks are carried along with the RTL as it is integrated into a system.
  – Maximize re-use of block-level verification efforts
• Formal, Simulation, & Hardware Usage
  – Use in both static and dynamic environments maximizes the chance of finding errors
  – Portable to emulation and FPGA’s
ABV Challenges SW and HW

• Intent
  – Understanding and integrating assertions at intended part of the design
  – How easy and quick is it to correct mistakes?

• Statistics
  – Getting the overall metric report from the run.

• Debugging?
  – Why did the assertion FAIL …

• Speed
  – For large designs, slow run time for simulation/Emulation

• Resource
  – Capacity constraints on FPGA’s used for Prototyping
Assertion checks using FSDB

Traditional

- Vendor independent FSDB format
- Supports all SVA types
- Easy to Debug
  - Source/Waves/Statistics
Assertions in Simulation
Synthesizable assertions in Emulation

- Subset of the SVA language is synthesizable
- SVA can consume lot of hardware resources
- Live-Processed (like simulator) or Post processing available
Zebu Emulation: Simulator-like Debug

- Full visibility (RTL & gate level)
  - All registers, nodes, memories
  - Run-time assertions control
  - No recompiles required
- Open standard support
  - FSDB, VCD, etc.
Protolink: Full visibility from Prototype
Protolink Advance Probes

Assertion-based

- Create a probe based on an assert property
- Merge assertion triggers on the PLC
Protolink Advance Probes – Assertion based

Simulation result:

ProtoRun dump:
Probe shows ‘high’ when the assertion fails
Advanced Assertions Debug Platform
Assertion Statistics

- Overall assertion result view for the simulation run
  - Support Multi-FSDB
  - Gives you a hint for the design quality
- Detail information for the property
  - Check assertion fail/success along the time
  - Sync with waveform
  - Quick access to Assertion Analyzer
  - Text report for further re-use
Assertion Analyzer

- Industry leading assertion debugging environment
- Intuitive way to debug assertion fail
- Analyze sub expressions in assertion statement
- Display variable value with different time on the same expression
- Insert related waveform into waveform window
- Direct link to HDL debug environment
Post processing Assertion Checks

- **Post processing of assertions on functionally correct events database (FSDB)**

- **Speed**
  - no need to re-compile design
  - Check low-level “implementation” assertions with short TAT

- **Flexibility—Vendor Independent**
Assertion Manager

- Assertion Manager
  - Manage assertion in the design
  - Table View
  - Hierarchy Tree View
  - Control which assertion need to be checked
  - Active and inactive number for assertion check
Post processing Assertion Evaluator

- Post processing Assertion Check
  - Reduce simulation time
  - Quick way to get assertion result
  - Check the assertion need to be checked
  - Transparent Virtual FSDB – No database merge needed
Complete Assertion Debug Flow

- Verdi provides complete Assertion solution
  - From assertion evaluate to HDL debug
  - A full debug loop from assertion to HDL
- Quicker assertion eval. Engine
  - No re-simulate needed
  - Shorten assertion develop time
  - Quicker debug cycle
- Better assertion visualization
  - Help to analyze assertion
  - Locate the problem quicker and easier
Conclusions

• Writing assertions finds spec ambiguities
• Methodology transparent
• Having a common platform for analysing and Debugging ABV methodology is a huge productivity boost
• Speed up in iterations time in qualifying new assertions can be achieved using the post processing methods