Hardware Functional Verification – Present and Future

HVC ‘2013

Yuval Caspi
Agenda

- Evolution of Functional Verification
- Languages and Methodologies Timeline
- Advanced Techniques Description and Trends
- Short Case Study
- 4 Challenges in Verification
- Future Verification Trends
“testing shows the presence of bugs not their absence”

Edsger W. Djikstra
Verification Evolution is driven by SOC Design

- Increasing Complexity
  - Continuous increase in number of IP's and embedded processors
    - 2006: 30-40 IP's, 1 CPU
    - 2011: 80+ IP's, 6+ CPU's
    - 2014: 120 IP's, 20+ CPUs?
- Exploding State spaces
- Increasing number of functions
  ... but ...
  - Reduced timeframe
  - Same Resources
Evolution of Functional Verification

- Architecture
- Micro-Architecture
- RTL-level Model
- Hand-crafted Test patterns
- Simulation

Manual, labor intensive, too expensive for increasing complexity
Evolution of Functional Verification

- Architecture
- Micro-Architecture
- RTL-level Model
  - Chip
  - Unit
- Testcase Generator
- Test patterns

Covers only small subset of total state space, often finds one bug in a problem area but not all related ones.
Evolution of Functional Verification

- **Architecture**
  - Manual definition of rules, limited to small design pieces

- **Micro-Architecture**

- **RTL-level Model**

- **Testcase Generator**

- **Testpattern**

- **Formal Verification**
  - Formal Rules
  - Simulation
Evolution of Functional Verification

High effort for environment setup and further design complexity increase

Architecture

Micro-Architecture

Testcase Generator

Test patterns

Coverage Models

Statistics

Analysis

Formal Verification

Formal Rules

RTL-level Model
Evolution of Functional Verification

- Architecture
- Testcase Generator
- Test patterns
- Coverage Models
- Statistics
- Analysis
- Formal Rules
- Formal Verification
- High Level Model
- Simulation
- RTL-level Model
- Unit
- Chip
Evolution of Functional Verification

- Architecture
- Micro-Architecture
- High Level Model
- RTL-level Model
- Formal Verification
- Formal Rules
- Testcase Generator
- Testpattern
- Coverage Models
- Coverage Statistics
- Analysis
- Manual effort to reflect Coverage Analysis in testcase generation
- Simulation
Evolution of Functional Verification

- Architecture
- Micro-Architecture
- Testcase Generator
- Coverage Models
- Coverage Statistics
- Test patterns
- ACF Semi FV
- Formal Rules
- Formal Verification
- Leverage FV technology for system level verification

Leverage FV technology for system level verification
Evolution of Functional Verification

- Architecture
- Micro-Architecture
- Testcase Generator
- Testpattern
- Coverage Models
- ACF Semi FV
- Coverage Statistics
- High Level Model
- RTL-level Model
- Formal Rules
- Automatic FV
- Simulation
- Formal Verification
Languages and Methodologies Timeline

- HVL – Constrained Random Based
- Specman
- OpenVera
- SystemVerilog
- VMM
- OVM
- UVM
- CDV
- Intel/IBM
- ABV (FV)
- Academic
- SFV/HFV
- OVL
- PSL
- SVA

Advanced Techniques Description and Trends

- Coverage-Driven Verification (CDV)
- Formal Verification
  - Active R & D
  - Assertion-Based Verification (ABV)
  - Semi Formal Verification (SFV)
  - Automatic Formal Verification
Coverage-Driven Verification

- Scenarios are monitored by coverage directives
  - Need **much less** test cases
- Unified Coverage accumulation use coverage of simulation and formal tools
- Automatic Coverage Feedback (ACF)
  - Automatic analysis of coverage results alters the test constraints to hit coverage holes
Coverage Techniques
Trends

Verification Techniques
Dynamic verification trends

<table>
<thead>
<tr>
<th></th>
<th>2007</th>
<th>2012</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code coverage</td>
<td>48%</td>
<td>70%</td>
</tr>
<tr>
<td>Assertions</td>
<td>37%</td>
<td>63%</td>
</tr>
<tr>
<td>Functional coverage</td>
<td>40%</td>
<td>66%</td>
</tr>
</tbody>
</table>

Non-FPGA Study Participants

Wilson Research Group and Mentor Graphics, 2012 Functional Verification Study. Used with permission

* Multiple answers possible
Formal Verification – Active Research and Development

- Assertion-Based Verification
  - Model Checking
  - Bounded Model Checking (BMC)
  - SAT Solvers
  - Symbolic Simulation
  - All Commercial tools use some or all of these techniques
- Semi-Formal Verification
  - IBM SixthSense, Synopsys Magellan
- Automatic Formal Verification
  - Jasper Gold, Cadence Incisive
Assertion-Based Verification

- Bus Protocols
- Block level properties
  - Designers play a significant role!
  - Invariants (always, never, etc.), Arbiter, FIFO, Complex properties
- Arithmetic functions
- Hierarchical verification by using abstractions (black-boxing)
  - Inter block handshake protocols
Semi-Formal Verification

- Using formal-techniques aiming at uncovering bugs (finding a counter-example) instead of achieving a complete proof of all the properties
- Critical for “hard-to-get” bugs
- Key for scaling formal techniques to large designs
Automatic Formal Verification

- Connectivity path verification (at the full-chip level)
- X propagation
- Deadlock detection
- Security violations
Formal Verification Status

Verification Techniques
Static verification technique trends

- Formal verification - property or model checking: 27%
- Automatic formal verification - super linting using formal techniques (e.g., x checking, deadlock detection, etc.): 13%

Wilson Research Group and Mentor Graphics, 2012 Functional Verification Study, Used with permission

HF - January 2013 Master Set, WRG & MG Study Results
Verification Techniques

Formal property checking adoption by design size

Formal Property Checking Adoption by Design Size

(Gate Count Excluding Memories)


HF - January 2013 Mentor Set, WRG & MG Study Results
ABV Languages Status

Design and Verification Languages
*Trends in languages and libraries for specifying assertions*

<table>
<thead>
<tr>
<th>Assertion Languages and Libraries</th>
<th>2010</th>
<th>2012</th>
<th>Next Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>OVL</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SVA</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PSL</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Synopsys OVA</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OTHER</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Wilson Research Group and Mentor Graphics, 2012 Functional Verification Study, Used with permission

*Multiple answers possible

HP - January 2013 Master Set, VRG & MG Study Results

© 2012 Mentor Graphics Corp. www.mentor.com
Case Study

- Direct Memory Access (DMA) Core Verification
- 2 Interfaces with different clock speeds
  - Slow interface for host access
  - Fast interface for both descriptors read and the memory access (Data Transfer)
- Multiple active DMA channels with weighted Round-Robin
- Supports Out-of-order response for Data read and write
- Completion notification should be in-order
- Interrupts are used to notify completion and errors
DMA Verification Environment
DMA Verification Key Points

- Correct Behavior Checking
  - Registers Access
  - Data transfer
  - Completion Interrupts ordering
  - Error Interrupts
  - Performance with all possible slave latencies and OOO responses

- Coverage Directives
  - All control and status register fields exercised
  - Out-of-order is exercised
  - Clock-domain crossing contentions
  - All configuration legal combinations
  - Back-pressure scenarios
  - Performance parameters
Methodology

- UVM with in-house guidelines
- All functionality is implemented in the environment
- Use Macros (logging and factory)

Test writing

- When possible, use only factory overrides
DMA Verification Environment

- Did I forget anything?
- Formal Verification
  - Arbiters
  - FIFOs
  - Bus Interfaces
  - Handshake protocols
- Automatic Coverage Feedback
  - Test cases can be altered with additional constraints extracted by coverage analysis tools
- Semi-Formal Verification
  - Light-houses: Use simulation to reach high risk states and then use formal techniques to cover the state space around it exhaustively
DMA Environment Integration to Higher Hierarchies

- Whenever possible use bus monitors and other interfaces (instead of looking at environment constructs)
  - Coverage/Interrupt checker should obey this guideline
- Host bus interface VIP could be replaced by the actual host
  - Host registers access sequences should be replaced by SW/FW that performs the same functionality as the sequences
  - Configurations are reconstructed by monitoring the host bus
- Fast bus interface VIP could be replaced by the actual slave architecture
  - Affects end-to-end scoreboard – different memory models
- This concludes are DMA case study
4 Challenges in Verification
4 Challenges in Verification

- Mastering Verification Complexity
  - Continuous increase in number of IP’s and embedded processors
    - 2006: 30-40 IP’s, 1 CPU
    - 2011: 80+ IP’s, 6+ CPU’s
    - 2014: 120+ IP’s, 20+ CPU’s
  - The more IP’s the higher the risk of late spec & implementation changes
  - Driving towards true Hw/Sw Co-Verification
  - Reuse of verification environments / stimulus from IP-level into big multi-CPU SoC environments
  - Reuse across projects
4 Challenges in Verification

- **Completeness**
  - When are we done?
    - Number of tests, Coverage results, Bugs convergence curve
  - Specification/Requirements Based Verification
    - Use e.g. UML (Unified Modeling Language) to write requirements and testplans
    - Is it feasible?
  - Review process
    - How many reviews are conducted?
    - What are the contents of each review?
  - Standard Checklists
Tests and Regressions

Verification Techniques
Number of tests created to verify the design

Verification Techniques
Regression Time

Wilson Research Group and Mentor Graphics, 2012 Functional Verification Study, Used with permission

HP - January 2013/HPA Set, VRG & MG Study Results
Faults

Effort and Results
Trends: Types of Flaws

Effort and Results
Trends: Root Cause of Functional Flaw in Non-FPGA
4 Challenges in Verification

- **Mixed Signal Verification**
  - Simulation performance improvement
  - Simulation model creation
  - Unified verification methodology covering analog and digital design parts
  - Verifying logic connectivity including low-power verification
  - Common language between analog and digital/verification engineers
  - Accurate power analysis
4 Challenges in Verification

- **Productivity**
  - Clean integration between formal techniques and simulation
  - Unified coverage database
  - Faster debug
    - New advances in simulators should become ubiquitous
  - Post-Silicon Debug acceleration
Future Verification Trends

- Sharing of Verification IP (Reusability)
  - Integration becomes seamless
  - C Code and HVL sequences are purely interchangeable
- HW/SW Co-verification
- Integration of Analog world into functional verification
- Formally verifying security features, firmware (Bob Bentley Intel)
- Do symbolic analysis of analog circuits (Bob Bentley Intel)
- **Approaching 20 years since the inception of the e Language (1st HVL)**
  - Are we on the verge of a new era?
Summary

- We presented:
  - Functional Verification Flow
  - Present Verification techniques and methodologies
  - A case study of a DMA
  - 4 Challenges in today’s Verification activities
  - Future Trends
Thank You