Viewing Formal Through Simulation-Tinted Glasses
TVS Formal Day 2016

Christian Burisch
Director Application Engineering
The Potential of Formal-Sim

Simulation and Formal Powerful Combination

- The effect of combining techniques in the flow

“Connection Points” to Improve the Flow

- Looking at planning, coverage, assertions/stimulus

OneSpin Pioneering Formal-Sim Integration

- Latest work by OneSpin in this area
Formal-Sim: Leveraging Orthogonality
Higher Coverage, Achieved Faster

A design transitions between many states during operation.

Formal checks all possible states over set time to exhaustively test properties.

Simulation checks one sequence of states driven by stimulus to replicate operation.
Simulation-Formal Combined Advantages

**Simulation** | **Formal**
--- | ---
Normal Operation | Complex scenarios, All combination analysis
Data path | Control path, data transport
High clock cycle depth, single state sequence | Shallow clock cycle depth, all states considered
Show correct behavior | Find unanticipated affects
Increasingly, Formal Layered Into Simulation Environments

- Formal being used where simulation falls short
  - Critical components that must be exhaustively verified
  - Corner cases with complex scenario stimulus setup
  - Blocks with many operational permutations
  - Automated analysis tasks

- However, methodologies still very disconnected
  - OneSpin working on ideal methodology
Formal Users Today
Disconnected from simulation process

Formal tiered use models

- **Formal Experts**: Complex assertions for key verification tasks
- **Designers & Verification Specialists**: Some assertions where necessary
- **General Engineering**: Automated apps

Disconnected from Simulation

- **Verification Planning**
- **UVM Testbench**
- **Simulation**
- **Debug**
- **Coverage DB**
Tearing Down The Verification Wall
Recipe for streamlined verification methodology

- Inclusive verification planning
- Common coverage models
- Connected debug environments
- Consistent/connected stimulus
- Formal with simulation “familiarity”
Verification Planning
Consider Formal early in the process, not only at end

Possible Formal Test Scenarios
- Are virtual registers connected correctly to real registers?
- Key block: ensure correct operation under all scenarios
- Does error correction always produce the correct result?
- Ensure critical signal (Fm Align) occurs with right timing
- Can internal FIFO ever overflow?
OneSpin Quantify™ Observation Coverage

Precise Coverage Closure

- Most precise formal coverage algorithm available
  - Checking for observation as well as controllability
- Assertion development guidance towards closure
  - Reduces redundant testing, shows uncovered areas
- Detects unreachable code
  - Over constrained, unreachable code
- Similar model to simulation code coverage

**Observation Coverage** more precise and efficient than other “Mutation” or “Cone-of-Influence” -based methods
Aligning Formal Coverage with Simulation

Common models recognizable to simulation users

- Assertions can be aligned with functional coverage from verification plan
- Need to check formal code coverage, aligned with simulation code coverage
- Need common coverage database to feedback results into planning tool
  - Accellera UCIS

![Diagram]

Assertion coverage with a similar model to simulation code coverage

- Code covered by assertions
- Code not covered by assertions
- Code that could never be covered
Available Today: Test Generation From Formal Traces

Design → Code → Execution Trace → Auto-Generate Test

```verilog
always_ff @(posedge clk or negedge reset_n)
begin
    if (!reset_n) begin
        PREADY <= 1'b0;
        PSLVERR <= 1'b0;
    end
    else if (PSEL && !PENABLE) begin
        PREADY <= 1'b1;
        PSLVERR <= !(PADDR inside
                   {cfg_perf_cnt_addr,
                   cfg_non_cacheable_addr});
    end else begin
        PREADY <= 1'b0;
        PSLVERR <= 1'b0;
    end
end
```
Expert User Providing UVM Tests
Want assertions to be usable in regression

- Research on methods to convert assertions to UVM tests
- Formal assertions hard to include simulation regression
  - Useful to show verification team testing performed
  - Valuable to feedback issues to designers for debug
  - Formal testing re-run without formal tool in regression
Abstract Assertions: More “Simulation-Like”
Align Operational Assertions with UVM Sequences

• Simplify assertion creation, closer to specification
• OneSpin SystemVerilog Library – industry standard
  • One change required for its use with other tools
• Allow timing diagram style assertions
  • Reduce low-level detail, like UVM sequences

```verilog
property my_prop
    state == busy && transfer == 1
implies
    ## 1 (read_acc && !write_acc) [*2]
    ## 1 (!read_acc && write_acc) [*2];
endproperty
```

<table>
<thead>
<tr>
<th>state</th>
<th>busy</th>
</tr>
</thead>
<tbody>
<tr>
<td>transfer</td>
<td>1</td>
</tr>
<tr>
<td>read_acc</td>
<td></td>
</tr>
<tr>
<td>write_acc</td>
<td></td>
</tr>
</tbody>
</table>
OneSpin Formal-Sim

- High-performance/capacity via advanced platform
- Observation Coverage: precise closure metric
- Operational Assertions: abstract assertion authoring
- Integrated flow with simulation friendly use model
  New partnerships with Synopsys and Mentor
Powerful Solutions Require Strong Technology and Apps

Unique, Powerful Verification Solutions

### Advanced Verification Solutions
- Metric-Driven Verification
- Block Integration Validation
- FPGA Implementation Verification

### Innovative Verification Solutions
- Agile Design Evaluation
- Safety Critical Verification
- SystemC/C++ Design Verification

Broad Range of Automated Apps

Automated Inspection | Design Exploration | Sequential EC RTL-RTL | Assertion Verification | Operational Assertions | Observation Coverage | DV Apps

- X-Prop
- Connect
- Register
- Scoreboard
- VIP
- Activation

SystemC/C++ Arithmetic | Sequential EC FPGA | Fault Qualification | Security Verification | Specification Verification

High Performance, Comprehensive Technology Platform

Proof Engines | Formal Model | Adv. Debug | LaunchPad

SystemVerilog . VHDL . SystemC . SVA . PSL
Formal-Sim Summary

Simulation and Formal Powerful Combination
• Orthogonal verification techniques
• Together can increase coverage and reduce effort

Several “Connection Points” Will Improve Flow
• Verification planning including formal
• Common coverage model
• Common debug (by and large solved issue)
• Familiar assertion authoring methods

OneSpin Pioneering Formal-Sim Integration
• Observation Coverage + UCIS integration
• Operational assertions and UVM connection
• Simulation integration models

For more information please visit
www.onespin.com