Test and Verification Solutions

Experts in Verification!

Virtual Platform Verification

Mike Bartley, TVS
Agenda

• **Distinguish Validation from Verification**
  – We will focus on verification

• **Identifying the golden reference**

• **Comparing Virtual Prototype vs. RTL**
  – Can we use formal equivalence?
  – What are the main issues with dynamic
  – Communicating between SV (System Verilog) and SC (SystemC)
Verification vs. Validation

- **Verification:**
  - The process of evaluating a product to determine whether the output of a development phase satisfy the conditions imposed at the start of that phase. [IEEE-STD-610].

- **Validation:**
  - The process of evaluating a product to determine whether it satisfies specified requirements. [IEEE-STD-610]

- **Validation = building the right product**
- **Verification = building the product right**
Verification vs. Validation

Market Requirements

Transformation

Validation

Software Engineers

Stakeholders

GOLDEN REFERENCE
Higher Abstraction Level

IMPLEMENTATION
Lower Abstraction Level

How much automation?
Verification Independence

- Verification must be independent from implementation
  - Designers and Verification engineers both interpret the specification

Verification relies on both not making the same interpretation mistake!
Verification vs. Validation

Market Requirements

Validation

Software Engineers

Stakeholders

Transformation

GOLDEN REFERENCE
Higher Abstraction Level

IMPLEMENTATION
Lower Abstraction Level

Verification

Traditionally = Natural Language

Traditionally = RTL
Verification vs. Validation

Market Requirements

Software Engineers

Stakeholders

Virtual Prototype in SystemC

Still RTL

Validation

Transformation

GOLDEN REFERENCE
Higher Abstraction Level

IMPLEMENTATION
Lower Abstraction Level

Requirements

Virtual Prototype in SystemC

Still RTL
• We need to **VALIDATE** the virtual prototype
• And **VERIFY** the implementation against it
Virtual Prototype as Golden Reference

- How much automation in the verification?
  - Can we use formal verification?

- IO matching
- State Matching
- Combinatorial equivalence

Implementation
- RTL
- C/C++/SystemC

Reference Model
- C/C++/SystemC

Equivalence Checker
- RTL Model
- Transformed RTL

100% Coverage
Using Formal Verification on a Virtual Platform

• **Normal Equivalence checking does not work**
  – It relies on state equivalence and combinatorial checks
    • Some tools can verify retiming
Are the two circuits below equivalent?

Yes: with State Negation (state \( f_2 = \neg f_4 \))
Using Formal Verification on a Virtual Platform

• **Normal Equivalence checking does not work**
  – It relies on state equivalence and combinatorial checks
    • (some tools can verify retiming)

• **Need to use sequential equivalence checks**
  – Demonstrate that the outputs are the same
    • (cannot use equivalent internal state to reduce the problem to combinatorial equivalence)
  – Tools are emerging
Hector from Synopsys

DFG = Dataflow Graph

SAT = Satisfiability Solvers
SMT = Satisfiability Modulo Theories
BDD = Binary Decision Diagram
• **Normal Equivalence checking does not work**
  – It relies on state equivalence and combinatorial checks
    • (some tools can verify retiming)

• **Need to use sequential equivalence checks**
  – Demonstrate that the outputs are the same
    • (cannot use equivalent internal state to reduce the problem to combinatorial equivalence)
  – Tools are emerging
  – Cycle timing differences add an extra complication
  – Add result accuracy (see paper from Imagination)
### Reduced Accuracy Function Verification

(Imagination paper on Hector presented at SNUG)

<table>
<thead>
<tr>
<th>Single Precision Floating Point Multiplier</th>
<th>Delay (ns)</th>
<th>Area (μm²)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Round Towards Zero</td>
<td>1.024</td>
<td>8208</td>
</tr>
<tr>
<td>1 unit in last place (ulp)</td>
<td>0.979</td>
<td>6057</td>
</tr>
</tbody>
</table>

**Diagram:horse:**

- **Representable Numbers**
- **Round Towards Zero (RTZ)**
- **Correct Result**
- **Acceptable 1ulp Results**
Virtual Prototype as Golden Reference
Using a Dynamic Verification Approach

Transformation

GOLDEN REFERENCE
Virtual Prototype

IMPLEMENTATION
RTL

Verification

• How much automation in the verification?
  – Can we use dynamic verification?

Stimulus

C/C++/SystemC Model

RTL Model

Comparison

How much coverage?

Cycle accuracy?
Connecting SystemC Models into your UVM test bench

- Sequencer generates data items to send into the DUT
- If the DUT is in SystemC then we can pass data across the DPI
- Or through TLM

```plaintext
task drive_item (input simple_item item);
begin
    SendStartToVirtualIP ( );
    SendDataToVirtualIP  (item.data);
    SendStopToVirtualIP ( );
    #50ns;
end
endtask : drive_item
```

Avoid hard coded delays
Use events where possible
SystemC TLM is wrapped to be pin compatible
- `SC_METHODs` to translate incompatible types
- BFM to translate transaction level

- Mentor UVMConnect implements TLM
- And also provides a UVM Command API
  - For accessing and controlling UVM simulation from SC
Good Resources for TLM

• Mentor UVM Connect
  – www.mentor.com/products/fv/events/uvm-connect

• Doulos
  – www.doulos.com/knowhow/systemc/tlm2/

• Accelera
• **Specification**
  – IPX is a fairly complex timer.
  – It supports multiple modes and runs on certain external events or an external clock which can be scaled.
  – It incorporates capture and compare logic and also provides interrupts for the application layer.
  – It operates in two clock domains, with one clock used for register configuration, driving output signals & interrupts and the other clock is used for timer operation and updating a few registers.
• **Timer model had to be cycle accurate**
  – AHB interface was used to configure the timer
  – Internal registers get updated before the acknowledgement at the bus peripheral interface.
  – Probe internal RTL signals to capture the exact time when the registers get updated

• **SystemC wrapper does the RTL to TLM and vice versa conversion for the AHB interface**
  – But not cycle accurate
  – resolved by generating internal events from systemC model
TVS Case Study 3: CPU model verification

- **Differences between RTL and C model**
  - RTL has a dual issue pipelined implementation
  - C model has single issue, non pipelined

- **Comparison points must be determined**
  - At end of simulation
    - Debug too hard
    - Perform comparison on instruction retirement
      - Need to access internal RTL signals

- **Interrupts and events**
  - Timing differences lead to false failures

- **Bus traffic generation**
  - Use BFM to generate bus traffic
Virtual Prototype NOT as Golden Reference

- We need to **VERIFY** the virtual prototype
- Do we **VERIFY** the implementation against it?
• Dynamic verification by simulation
• There is no executable reference model
Verification vs. Validation

- Verify the RTL against the Virtual prototype
- The software developers will verify too
Summary

• Is your Virtual Prototype the golden reference?
• Virtual Prototype vs. RTL
  – Formal requires sequential equivalence
    • Only some solutions available
  – Dynamic verification is the main solution
    • TLM, UVM Connections
    • Data Accuracy
    • Cycle Accuracy