How to Kill 4 Birds with 1 Stone: Using Formal Verification to Validate Legal Configurations, Find Design Bugs, and Improve Testbench and Software Specifications

Saurabh Shrivastava, Kavita Dangi, Darrow Chu, Mukesh Sharma
Motivation

How to get a head start on verification

How to augment simulation based verification

How to validate/verify device configuration for
  – Better testbench simulations
  – Better software output

How to weed out basic bugs:
  – Bus contention, x-assignments, floating-bus, FSM

Assign out = SEL[0] ? in0 : 1’bz;
Assign out = SEL[1] ? in1 : 1’bz;

Case(sel)
  0: out = in0;
  2: out = in1;
  default: out = 1’bx;
Older Approach

- Block2 TB
- Block3 TB
- Block1 TB
- Full Chip TB
- BlockN TB

CFG

Configuration
Bus Contention
X - Assignment
FSM
Bus Floating Multidriver
Limitations of Older Approach

- Bugs are discovered late
- Block and Chip level simulations are as good as stimulus
- Difficult to hit all scenarios due to:
  - Big configuration space (only handful are verified)
  - Big design
- Interesting bugs found when test benches are sophisticated!
**New Approach**

- Checkers for illegal config
- Converted to Assume Constraints

**SVA Lib**

- **Block1 RTL**

**White Formal TB**

- **Better TB Checkers**
- **Better Software Output**
- **Bus Floating Multidriver**

**Targets:** Bus checks, X-assignments, range-checks, FSM-checks, etc

- Assertions are generated by the tool

**Configuration**

- **Bus Contention**
- **X - Assignment**
- **FSM**
New Verification Flow

White Formal Testbench
- Verify Constraints: Correct & Complete
- Find design bugs early
- SVA Lib
- Block RTL

Block/FullChip Testbench
- Used as Checkers
- Catches Incorrect Block programming
- SVA Lib
- Block RTL
- Find design bugs

Software
- SVA Lib
- Block RTL
- Prevents incorrect IP programming
- Read by SW
- End User IP Configurator

Sponsored By: accellera SYSTEMS INITIATIVE

© Copyright 2013 Xilinx
**Assertion Library Example**

**fileA_assert_lib.sv**

```verilog
reg [3:0] varA;
reg [5:0] widthA;
reg modeA;

`LIB_ASSERT_ONE_HOT(22,"varA value is not compliant",varA)

`LIB_ASSERT_CHECK(23,"Incorrect width for modeA=0",modeA==0 && widthA==16)
```

**assert_lib_define.svh**

```verilog
`define LIB_ASSERT_ONE_HOT(INST,MSG,a) 
`ifdef FORMAL \ 
  assume_one_hot_``INST``: assume property (@(posedge clk) $onehot0(a)); \ 
`else \ 
  assert_one_hot_``INST``: assert property (@(posedge clk) $onehot0(a)) \ 
  else begin $error("%s: one_hot fails for a=%h\n",MSG,a); \ 
`endif
```

**run.log**

```
ncsim: *E,ASRTST (lib_assert_macros.sv,17): (time 17 NS) Assertion 
top.dut.modA.assert_one_hot_22 has failed 
varA value is not compliant: one_hot fails for a=b
```
Bugs Found

- Incorrect library assertion causing bus contention
  
  Assign out = SEL[0] ? in0 : 1’bz;
  Assign out = ~SEL[1] ? in1 : 1’bz;
  `LIB_ASSERT_CHECK(sel,"Select values are incorrect", (SEL==0 || SEL==1 || SEL==6 || SEL==10))

- Combinatorial loops

- Range overflow

- Missing library assertions causing
  - bus contention
  - X-assignment failures
  - Bus floating failures

- State reachability and transition

- State deadlock issues

- Bus floating failures
<table>
<thead>
<tr>
<th></th>
<th>Old Approach</th>
<th>New Approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>Early bug discovery</td>
<td>![Hand Down]</td>
<td>![Hand Up]</td>
</tr>
<tr>
<td>Exhaustive</td>
<td>![Hand Down]</td>
<td>![Hand Up] ![Hand Up]</td>
</tr>
<tr>
<td>Setup time</td>
<td>![Hand Down] &gt; 1 Month</td>
<td>![Hand Up] 1-3 days</td>
</tr>
<tr>
<td>Setup effort</td>
<td>![Hand Down]</td>
<td>![Hand Up] ![Hand Up]</td>
</tr>
<tr>
<td>Test coding</td>
<td>![Hand Down]</td>
<td>![Hand Up] ![Hand Up] ![Hand Up]</td>
</tr>
<tr>
<td>Scope</td>
<td>![Hand Up]</td>
<td>Targeted ![Hand Down]</td>
</tr>
<tr>
<td>Configuration verification</td>
<td>Manual approach, limited</td>
<td>Does not verify anything out of cone of logic, but verifies ALL legal cfgs for properties</td>
</tr>
<tr>
<td>Tool issues</td>
<td>![Hand Up]</td>
<td>![Hand Down] ![Hand Down]</td>
</tr>
<tr>
<td>Tool learning</td>
<td>![Hand Up]</td>
<td>![Hand Down] ![Hand Down]</td>
</tr>
</tbody>
</table>

> Only opportunistic!
Future Work

▶ Multi-block verification
▶ Tool kit
▶ Enable designers to run the flow
▶ Gate level Verilog verification
Summary

- Finding early design bug and verifying configurations is an issue
- Simulation based approaches have limitations
- White Formal is used along with SVA library to solve this problem
- New approach finds early bugs and verifies configuration space
- As a result we get better testbench and better software output
White Formal

X-assignment example
Older Approach

- Block and Chip level testbenches
- Directed & pseudo-random tests for:
  - bus contention, X-assignment, FSM-checks
- Static lint for bus floating and multi-driver
- Configuration verification
  - Manual
  - Directed simulations
Limitations of New Approach

- Only opportunistic!
- Works only for targeted checks
- Configurations outside cone of logic not verified
- Tool capacity: cannot be applied at chip level
- Tool issues
- Tool learning