Speed up Emulation Debugging using White-box Assertions

Ricky Wang
ricky@atrenta.com
Verification Efficiency Is All About Debug

Average Time Spent on Verification Tasks

- Debugging 42%
- Developing Testbenches 26%
- Writing/RUNning Tests 26%
- Identifying Bugs 25%
- Managing Bug Dependencies 17%
- Other 6%

2013 independent survey, IC Manage

#1 Emulation Challenge: Debug
Why Emulation Debug Is So Challenging...

System-level HW/SW SoC Emulation Run (2 hrs @ 2MHz)

14 Billion Cycles

Bug

System-level crash

- End to end checkers (if any) do not fire...
- Emulators have limited trace buffer size
- Error can occur billions of cycles before being detected

Debug Window is too small (1M = 0.007% of 14 Billion!)

So Now What???
We Guess!!!
Why Emulation Debug Is So Challenging...

System-level HW/SW SoC Emulation Run (2 hrs @ 2MHz)

14 Billion Cycles

- End to end checkers (if any) do not fire...
- Emulators have limited trace buffer size
- Error can occur billions of cycles before being detected

Pick a new trigger
Restart the emulator
And Wait...
Repeat, Repeat, Repeat
# Why Emulation Debug Is So Challenging...

**Bug**

**System-level HW/SW SoC Emulation Run**
- (168 hrs @ 4MHz)
- (24 hrs @ 4MHz)
- (2 hrs @ 2MHz)

## Emulation Facts:
- Advertised Performance: Up to 4MHz
- Average test runtime: 1-2 days
- Average runtime for HW/SW development: 1+ weeks
- Average time to debug on emulator: 2 weeks...

## Cost of Emulation Debug

<table>
<thead>
<tr>
<th>Annual HW Cost</th>
<th>1-Week HW Cost</th>
<th>2-Weeks HW Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>$250,000</td>
<td>$4800</td>
<td>$9600</td>
</tr>
<tr>
<td>$500,000</td>
<td>$9600</td>
<td>$19,200</td>
</tr>
<tr>
<td>$1M</td>
<td>$19k</td>
<td>$38k</td>
</tr>
<tr>
<td>$2M</td>
<td>$38k</td>
<td>$96k</td>
</tr>
<tr>
<td>$5M</td>
<td>$96k</td>
<td>$192k</td>
</tr>
</tbody>
</table>

Plus:
- Electricity, Cooling,
- Engineering time,
- Time-to-Market,
- Etc

---

Note: The information presented on this roadmap is for information purposes only and is not a commitment, promise, or legal obligation to deliver any material, code, or functionality and should not be relied upon in making a purchasing decision.
Why Emulation Debug Is So Challenging...

- End to end checkers (if any) do not fire...
- Emulators have limited trace buffer size
- Error can occur billions of cycles before being detected

System-level HW/SW SoC Emulation Run (168 hrs@ 4MHz)

Bug

2419 Trillion Cycles

System-level crash

Average time to debug on emulator: 2 weeks...

Note: The information presented on this roadmap is for information purposes only and is not a commitment, promise, or legal obligation to deliver any material, code, or functionality and should not be relied upon in making a purchasing decision.
What We Need: ID Bug At Its Source

System-level HW/SW SoC Emulation Run (2 hrs @ 2MHz)

14 Billion Cycles

Bug
System-level crash

- End to end checkers (if any) do not fire...
- Emulators have limited trace buffer size
- Error can occur billions of cycles before being detected

Identifying Bugs at Source Requires Embedded Assertions!

- Assertions are embedded
- Assertions detect errors automatically
- Assertions point directly to source of error
- Assertions are independent of checker quality
An assertion specifies a behavior of the system

• Assertions are primarily used to validate the behavior of a design
• Assertions can be used to provide functional coverage and to flag that input stimulus

Types

• **Assert**: to specify the property as an obligation for the design that is to be checked to verify that the property holds
• **Cover**: to monitor the property evaluation for coverage
• **Assume**: to specify the property as an assumption for the environment. Simulators check that the property holds, while formal tools use the information to generate input stimulus
• **Restrict**: to specify the property as a constraint on formal verification computations. Simulators do not check the property
Assertions Overview

- Assertions are used for simulation, formal and emulation

- Assertion used mostly to describe “bad” behavior (should not happen)
  - “ERROR” state should never be reached
  - The overflow signal should never go high
  - After the WAIT state, don’t go back to the READY until after 5 cycles...
  - Most common way to use SVA(System Verilog Assertion)
  - Formal tools can prove or disprove them

- Assumptions restrict the behavior of the design
  - “assume these inputs are mutually exclusive”... “assume my counter is onehot”
  - Needed to model protocols, complex relationships, etc.: all input to AMBA busses cannot change willy-nilly
  - Reduces the search space and makes the analysis easier for the formal tools
  - Over constraining with assumptions can lead to missed bugs
Solving Emulation Debug Challenges

**Assertion Requirements for Emulation Debug:**

- Assertions Must Be Comprehensive (White-box)
  - White-box assertions focus deep within the block
  - White-box assertions are easy to debug
  - Black-box assertions are often temporal (consume more capacity)

- End to end checkers (if any) do not fire...
- Emulators have limited trace buffer size
- Error can occur billions of cycles before being detected

**Identifying Bugs at Source Requires Embedded Assertions!**

- Assertions are embedded
- Assertions detect errors automatically
- Assertions point directly to source of error
- Assertions are independent of checker quality
 Assertions: Black-Box Vs. White Box

- **Black-box Assertion (property)**
  - Describe the end-to-end function, usually specific protocol
  - Not many properties, only in some key modules
  - Not easy to debug when it is triggered

- **White-box Assertion (property)**
  - Fine grain: every 10-100 RTL lines generate one property
  - Do not simply restate RTL
  - Are orthogonal to RTL
  - Mostly one or two clock-cycle properties (easy to understand)

```verilog
ce_cnt0_nxt == 4'h0
inside(fcl_fdp_next_ctx_bf_l, [4'h7:4'he])
onehot(thr_f_dec)
  inside(ce_cnt_rst, [4'h0:4'h1], 4'hf)
cam_vld_bf -> rdreq_bf
mutex(rdreq_bf, ifq_fcl_rdreq_bf)
mutex(itlb_access_done, rst_itlb_stv_l)
ifq_fcl_wrREQ_bf -> allow_ifq_access_icd_bf
nextthr_bf_buf[3] -> rst_tri_en
rise(rst_tri_en) -> thr_f_dec[3:1] == 3'h0
tlu_itlb_data_rd_g -> fcl_reset
tlu_itlb_invalidate_all_g -> fcl_reset
all_stallreq && rst_stallreq && switch_bf |-> fcl_reset
(~rst_stallreq_d1 &~switch_bf | arst_vld_f) && rst_tri_en |-> fcl_reset
```
White-box Assertion Example

Master | req  | rdy | Memory Controller

Memory | If(req)
busy <= 1'b1;
else if(rdy)
busy <= 1'b0;

Redundant Properties:
req |-> ##1 busy
(!req & rdy) |-> ##1 !busy

Orthogonal Whitebox Property #1:
req |-> !busy
This means new req is only asserted when rdy for previous req is asserted before.

Orthogonal Whitebox Property #2:
rdy |-> busy
When rdy is asserted, in the same cycle, bus is always busy. That is, rdy must follow a req.

Equivalent Black-box Property (difficult to read):
$rose(req) |=> !req throughout ##[1:$] (rdy & !req);
Solving Emulation Debug Challenges

Assertion Requirements for Emulation Debug:

• Assertions Must Be Comprehensive (White-box)
  • White-box assertions focus deep within the block
  • White-box assertions are easy to debug
  • Black-box assertions are often temporal (consume more capacity)

• Assertions Must Be Generated Automatically
  • Never enough time to write manually
  • Designers don’t think to target all corner-cases

End to end checkers (if any) do not fire...
Emulators have limited trace buffer size
Error can occur billions of cycles before being detected

Identifying Bugs at Source Requires Embedded Assertions!
• Assertions are embedded
• Assertions detect errors automatically
• Assertions point directly to source of error
• Assertions are independent of checker quality

Trigger Buffer (1M clocks)
Introducing BugScope

Automated Assertion Generation For Emulation

IP Development

Test Suite → Simulation → Automatic Assertion Generation → BugScope Property Database → HW/SW Co-emulation

RTL → BugScope

HW/SW Co-emulation

Emulation

Automatically detect integration errors

Automatically detect untested configurations (and bugs)

Methodology for Assertion Reuse in SoC (MARS)

Fully Automated Flow
How BugScope Generates White box Assertions

Step 1
ndb generation

wr_ptr==rd_ptr |-> empty
fifo_cnt : 0 to 3
fifo_cnt: 0 to 6
fifo_cnt: 0 to 11

Step 2
Property generation

fifocnt < 4'd12
!(rd && empty)
!(rd && wr)

Step 3
SVA/PSL generation

SVA module
!(rd && empty)
fifo_cnt < 4'd12
!(rd && wr)

Atrenta Confidential © 2014 Atrenta Inc.
Famous HD controller Company

57 assertions are fired in Emulation (Palladium platform), one RTL bug is caught!

- Fired assertions

```verilog
TIME[103348] NEXTOP WARNING: assertion xhfifo_ggeil1 in <tb_iguana.iguana_top_inst.IGUANA_CORE.xhfifo_inst> failed
TIME[103483] NEXTOP WARNING: assertion xhfifo_11h82d in <tb_iguana.iguana_top_inst.IGUANA_CORE.xhfifo_inst> failed
TIME[103783] NEXTOP WARNING: assertion fifo_2s27lb in <tb_iguana.iguana_top_inst.IGUANA_CORE.xhfifo_inst.rd_ofs_fifo> failed
TIME[103783] NEXTOP WARNING: assertion fifo_s0sb59 in <tb_iguana.iguana_top_inst.IGUANA_CORE.xhfifo_inst.rd_ofs_fifo> failed
TIME[103786] NEXTOP WARNING: assertion fifo_85mbch in <tb_iguana.iguana_top_inst.IGUANA_CORE.xhfifo_inst.rd_ofs_fifo> failed
TIME[103786] NEXTOP WARNING: assertion fifo_2ui7bh in <tb_iguana.iguana_top_inst.IGUANA_CORE.xhfifo_inst.rd_ofs_fifo> failed
TIME[103786] NEXTOP WARNING: assertion fifo_cirk7c in <tb_iguana.iguana_top_inst.IGUANA_CORE.xhfifo_inst.rd_ofs_fifo> failed
TIME[103900] NEXTOP WARNING: assertion fifo_e8lsr5 in <tb_iguana.iguana_top_inst.IGUANA_CORE.xhfifo_inst.rd_ofs_fifo> failed
TIME[104770] NEXTOP WARNING: assertion fifo_rl9a59 in <tb_iguana.iguana_top_inst.IGUANA_CORE.xhfifo_inst.rd_ofs_fifo> failed
TIME[107713] NEXTOP WARNING: assertion xhfifo_tjo944 in <tb_iguana.iguana_top_inst.IGUANA_CORE.xhfifo_inst> failed
```

- The 1st fired Assertion – Bug Caught

```verilog
// xhfifo_ggeil1 : mutex(wdata_read, tag_is_for_read[wreq_tag])
property xhfifo_ggeil1;
 @(posedge ocp_clk) disable iff( ~ocp_rst_n || before_reset)
 !(wdata_read && tag_is_for_read[wreq_tag]) !== 1'b0;
endproperty : xhfifo_ggeil1
assert_xhfifo_ggeil1 : assert property( xhfifo_ggeil1) else begin
$warning("TIME[%0d] NEXTOP WARNING: assertion xhfifo_ggeil1 in <tb_iguana.iguana_top_inst.IGUANA_CORE.xhfifo_inst> failed", $time);
$assertoff( 0, assert_xhfifo_ggeil1);
end
```
Real-World Example: Customer 2

**Famous Communication Chip Company in Taiwan**

- They use Zebu emulation platform from Synopsys;
- 59/1441 properties are fired in emulation;
- 3 important verification holes are found
  - All were missed in IP-level verification
  - All were corner-case scenarios that needed further testing;
Comprehensive RTL Platform

- Complete Technology
  - IP and SoC Signoff

- Reference Methodology
  - High impact, low noise

- Management Reports
  - Linked HTML

- Flexible Use Models
  - Batch, Tcl shell and GUI
Thank You!