

Practical Approach Using a Formal App to Detect X-Optimism-Related RTL Bugs

Shuqing Zhao, Shan Yan, Yafang Feng
Broadcom Corporation



Agenda

- Problem Statement
- Related Work
- Methodology Outline
- Case Study 1: Power Management Controller
- Case Study 2: Audio Processor
- Summary

X Sources

- What is X? Simulation logic state — unknown (0 or 1)
- X Sources that can potentially hide RTL bugs:
 1. Uninitialized variables—nonresettable registers
 2. Out-of-bound bus bit select or array access
 3. Explicit X assignments that are reachable
 - “Don’t care” used for logic minimization purpose
 - Modeling unknown, e.g., memory output or gate output with no power
 - Modeling error cases—should be replaced with assertions
 4. Power-aware semantics specified by UPF/CPF
- Less likely RTL bug sources:
 - Floating wires/ports, multiple drivers—detected by lint tools
 - Timing violations

Problem Statement

- X-Optimism in standard RTL simulation is dangerous and causes bug escape.

- X-Optimism: “optimistically” resolving X, not matching silicon behavior

always @(posedge clk or negedge rst_n)

if (!rst_n)

count <= 0;

else if (count_enable)

count <= count + 1;

- Problematic Verilog constructs: if, case, posedge, negedge

- Gate-level simulation is not adequate.

- It suffers from X-Pessimism: e.g., (a & ~a) should not be x.
 - It usually is a subset of RTL simulation coverage.
 - Bugs found at this late stage require costly resynthesis or ECOs.

Related Work

- RTL coding style to explicitly propagate X.
 - Readability problem.
 - It's not practical to intercept all combinations.
- 2-state simulation—randomize X to 0 or 1.
 - Complete coverage is not practical.
- Model checking using 4-state:
 - Suffers usual formal capacity limitation—bounded proofs.
 - Depends on completeness of assertions created manually by users.
- Model checking of X checkers is automatically inserted.
 - Basis of our approach

Formal X-Propagation App

- How the formal X-prop app works:
 - Environment input: clocks, resets, modes, X to input ports.
 - Analyzes/elaborates RTL into an internal “netlist.”
 - Automatically creates the X detection assertions on targets:
 - Clocks and resets
 - Output ports
 - Test condition for if/case statements
 - User-specified signals
 - Prove that X’s cannot propagate to targets.
 - A counter-example is generated if any assertion fails.
- Main issue: some assertions will get full proof and some will only get bounded proof.
- Our contribution: a methodology to improve ROI.

X Source-Driven Methodology

Coding

- Async reset, use assertions for X, avoid casex/casez, etc.

Scan for X's

- Lint, “Formal” reset analysis, Structural property analysis, X-assignment reachability

Formal App

- Where can X propagate to?

Debug

- Counter-examples for failed assertions

X-prop Sim.

- Cover unproved X assertions
- Power-aware simulation with UPF/CPF

Case Study 1: PM Controller

- Background
 - DUT: Critical power management controller for quad core app processor
 - Presilicon verification
- Process
 - Ran very comprehensive UVM constraint random simulation.
 - Used formal reset analysis to check for uninitialized flops.
 - Used the formal X-prop app to scan for targets X can propagate to.
- Results
 - Reset analysis found four flops that were not initialized by reset.
 - X-prop app detected one flop that caused a nasty bug in the state-machine transition after reset.

XPROP App GUI Report

X-Propagation Analysis Browser

Module for coe_a7_cdc

Instance	Result
coe_a7_cdc (coe...	398:6:6
u_isidle_sync...	
u_isidle_sync...	
u_isidle_sync...	
u_clamp_core0...	0:0:5
u_clamp_core1...	0:0:5
u_clamp_core2...	0:0:5
u_clamp_core3...	
u_cci_reset_s...	1:0:0
u_arm_reset_s...	1:0:0
u_armcore0_re...	0:0:15
u_armcore0_re...	0:0:13
u_armcore0_re...	0:0:9

```

7 module coe_a7_cdc
8   /*AUTARG*/
9   // Outputs
10  apb_prdata_cdc, cdc_nFIQ, cdc_nIRQ, cdc
11  arm_sys_idle_out, ACINACTM, fsm_power_
12  fsm_power_sw_enable_low_irdrop, fsm_cl
13  all_clk_is_idle, PWRCTLO, SPM_SOFTRESE
14  ATB_SOFTRESETN, cci_rstn, arm_rstn, SO
15  COREPOR3_SOFTRESETN, COREPOR2_SOFTRESE
16  COREPOR0_SOFTRESETN, L2_SOFTRESETN, ax
17  DEBUG2_SOFTRESETN, DEBUG1_SOFTRESETN,
18  CORE3_SOFTRESETN, CORE2_SOFTRESETN, CO
19  CORE0_SOFTRESETN, cdc_debug_bus, a7_del
20  L2RSTDISABLE, ClkgenScanOut, cdc_spare
21  // Inputs
22  cdc_resetn, cluster_rstn, apb_prdata_

```

session_0

SUMMARY

```

=====
Total Tasks      : 4
Total Properties : 368
  assumptions    : 0
    - approved   : 0
    - temporary  : 0
  assertions     : 368
    - proven     : 204      ( 55.4% )
    - marked_proven : 0      ( 0.0% )
    - cex        : 159      ( 43.2% )
    - ar_cex     : 0         ( 0.0% )
    - undetermined : 5       ( 1.4% )
    - unprocessed : 0       ( 0.0% )
    - error      : 0       ( 0.0% )
  covers         : 0
    - unreachable : 0
    - covered     : 0
    - ar_covered  : 0
    - undetermined : 0
    - unprocessed : 0
    - error       : 0
determined
[<embedded>] %

```

[<embedded>] %

Console | Lint Messages | Warnings / Errors | Proof Messages

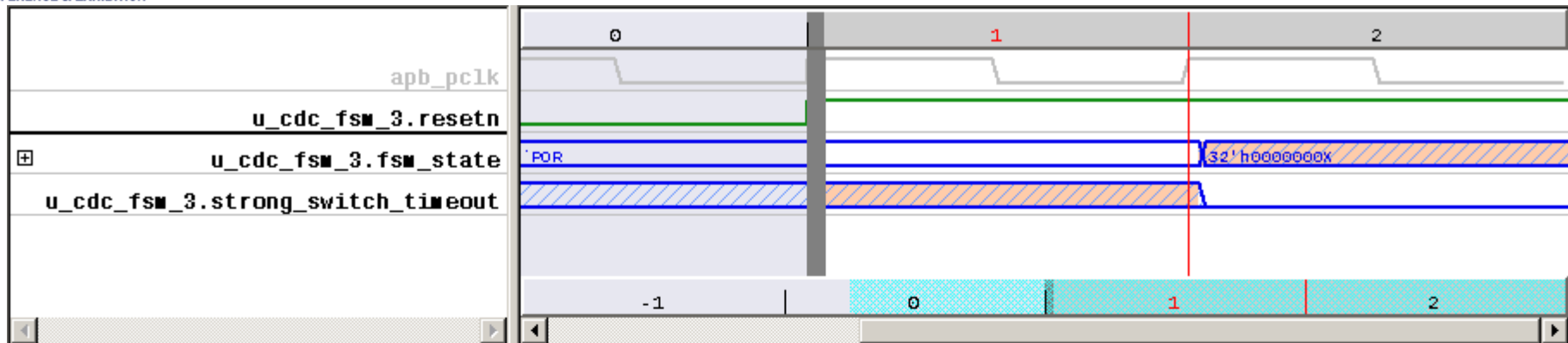
Task/Property Table

Name	Result
<embedded>	0:0:0
XP_clocks_and_resets	6:0:12
XP_control	186:5:121
XP_outputs	12:0:26

Filter on name: ab

Type	Name	Engine	Bound
✗	Assert(xpr... u_cdc_fsm_3__FD_RESET_TIMEOUT__L2_IS_ON_line_393...	B	43
✗	Assert(xpr... u_cdc_fsm_3__FD_RESET_TIMEOUT__FIRST_TO_POLL_thi...	B	43
✗	Assert(xpr... u_cdc_fsm_3__L2_IS_ON_cdc_power_ok_low_irdrop_s...	Ht	29
✗	Assert(xpr... u_cdc_fsm_3__L2_IS_ON_cdc_power_ok_low_irdrop_s...	Ht	29
✗	Assert(xpr... u_cdc_fsm_3__cdc_power_ok_inrush_limited_weak_sw...	Ht	22
?	Assert(xpr... u_cdc_fsm_3__wait_idle_timeout_line_436_col_13	N	1026 -
✓	Assert(xpr... u_cdc_fsm_3__STANDBYWFIL2_all_is_idles_asserted...	PRE	(0)
✓	Assert(xpr... u_cdc_fsm_3__STANDBYWFIL2_any_core_interrupt_pen...	PRE	(0)
✗	Assert(xpr... u_cdc_fsm_3__ARM_SYSIDLE_timeout_line_446_col_8	Ht	13
✗	Assert(xpr... u_cdc_fsm_3__CDC_BUSY_TIMEOUT_INT_interrupt_pen...	Ht	3
✗	Assert(xpr... u_cdc_fsm_3__fsm_state_line_244_col_13	Ht	2
✓	Assert(xpr... u_cdc_fsm_3__cdc_command_line_462_col_6	PRE	(0)
✓	Assert(xpr... u_cdc_fsm_3__cdc_command_line_464_col_6	PRE	(0)
✓	Assert(xpr... u_cdc_fsm_3__cdc_command_line_466_col_6	PRE	(0)
✓	Assert(xpr... u_cdc_fsm_3__cdc_command_line_468_col_6	PRE	(0)
✓	Assert(xpr... u_cdc_fsm_3__cdc_command_line_470_col_6	PRE	(0)
✗	Assert(xpr... u_cdc_fsm_3__cdc_cfg_fd_reset_timer__reset_timer_...	Ht	3
✗	Assert(xpr... u_cdc_fsm_3__cdc_cfg_cd_reset_timer__reset_timer_...	Ht	3
✗	Assert(xpr... u_cdc_fsm_3__fsm_state_line_484_col_92	Ht	2
✗	Assert(xpr... u_cdc_fsm_3__ARM_SYSIDLE_timer__cdc_cfg_armsysidl...	Ht	12
✗	Assert(xpr... u_cdc_fsm_3__fsm_state_line_525_col_8	Ht	11
✗	Assert(xpr... u_cdc_fsm_3__cdc_cfg_weak_switch_timer__fsm_state...	Ht	21
✗	Assert(xpr... u_cdc_fsm_3__cdc_cfg_strong_switch_timer__fsm_sta...	Ht	18
✗	Assert(xpr... u_cdc_fsm_3__state_changed_line_549_col_11	Ht	2

RTL Bug Counter Example



Source Pane

Search the Source ...

Why at iteration 1 for u_cdc_fsm_3.fsm_state (1

```

243
244     case (fsm_state)
245         *POR
246         *POR: begin //POR state
247             if (strong_switch_timeout) begin
248                 1'bx
249                 if (this_is_core_0)
250                     1'b0
251                     fsm_state <= *RESFDM;
252             else
253                 fsm_state <= *RESFD_WAIT; // no default here, go one way or the other
254             end
255         else fsm_state <= *POR;
256     end
257
258     *RESFDM: begin // Resume from full dormant as master (core_0 only)
259         if (FD_RESET_TIMEOUT & (cdc_command == *MDEC)) begin
260             fsm_state <= *RUN;
261         end
262     end
  
```

Case Study 2: Audio Processor

- Background
 - DUT: audio processor module
 - Post-silicon analysis of bug escape
 - Bug: one of the channels was hanging the system.
- Process
 - Gatesim reproduced the bug.
 - Used the X-prop app to scan for targets X can propagate to.
- Results
 - X-prop app confirmed out-of-bound array access due to an RTL coding error.
 - The SPS app caught the same bug in a shorter runtime.
 - Xprop app proved absolute safety of a software workaround.

Summary

- Conclusions
 - X-optimism causes RTL bugs to be missed from simulation.
 - Formal approach is exhaustive but has limitations.
 - We recommend an X source-driven formal approach.
 - We found show-stopper bugs using this approach.
- Future Plan
 - Run X-prop app + low-power formal app to detect X issues from UPF.
 - Study formal code coverage relationship for bounded proof.
- Acknowledgements
 - Jennifer Hwang from Broadcom

Thank you!