DVCon 2012

Design & Verification Conference & Exhibition

February 28 – March 1, 2012

X-propagation Woes: Masking Bugs at RTL and Unnecessary Debug at the Netlist

by

Lisa Piper

Technical Marketing Manager

Real Intent

Topics

- X-propagation Issues
- Customer Concerns
- Overview of Point Solutions
- Proposed Holistic Solution

What is an X?

- IEEE 1800 SystemVerilog
  - X is a value in 4-state logic (0, 1, X, Z) that represents an unknown logic value.
  - 4-state data types include: logic, reg, integer, time.
  - 4-state logic has a default value of X.
- Xs Do Not Exist In Real Hardware.
- Tools Interpret Xs Differently.

Tool	Interpretation
Simulation	Unknown: NOT 0 or 1
Synthesis	Don't Care: Either 0 or 1
Formal	Both 0 and 1

Where do X's come from

- **Uninitialized flops**
- Explicit assignments
- Bus Contention
- Out of Range
- Undriven
- Inputs
- Low power modes

X-optimism in RTL

```
always @(posedge clk)
  if(en) begin
     count = count +1'b1 ;
  end
```

```
always @(posedge clk)
  case(en)
     1'b0: count=count;
     1'b1: count=count + 1'b1;
  endcase
```

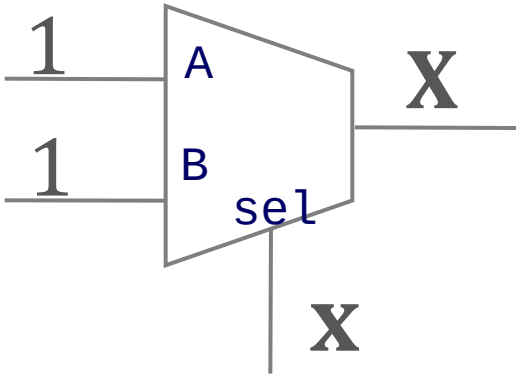
en	count
0	**count unchanged**
1	count = count + 1
X	**count unchanged**

X-optimism = unknown reduced to known

Can result in the masking of a functional bug.

X-pessimism in Netlist

A	B	Sel	Output
0	0	X	0
0	1	X	X
1	0	X	X
1	1	X	X



For A=B=1, sel = X:
output = A*sel + B*!sel
output = 1*X + 1 *!X
output = X + !X
output = X

X-pessimism = more X than necessary

False negatives mean more debug!

Example!

RTL

```
if (en)
      out = in1;
else
      out = in2;
```

Netlist Equivalent

$out = (en*in1) + (\overline{en}*in2)$

en	in1	in2	RTL out	Netlist out	Real World out
X	0	0	0	0	0
X	0	1	1	X	Undeterminable
X	1	0	0	X	Undeterminable
X	1	1	1	X	1

Optimism Pessimism

Two Distinct Problems

- Functional bugs can be masked in RTL simulation due to X-optimism.

- In netlist simulations there are many unnecessary Xs due to X-pessimism.

Both issues cause differences between RTL and Netlist.

Customer Concerns

- Customer #1 – Verification Engineer
  - Need RTL detection of X-optimism
  - Only report real functional issues
  - Use my existing infrastructure
  - Debug information is needed to find root cause

- Customer #2 – Design Engineer
  - Show me the X-sources in the design
  - Show me which constructs are susceptible to X-propagation
  - No automatic re-coding!

Customer Concerns

- Customer #3 – Verification Engineer
  - Primary concern was Xs from low power
- Customer #4 – Verification Engineer
  - Pessimism correction is critical
  - Need to get netlist simulations up and running quickly

X-Verification Point Solutions

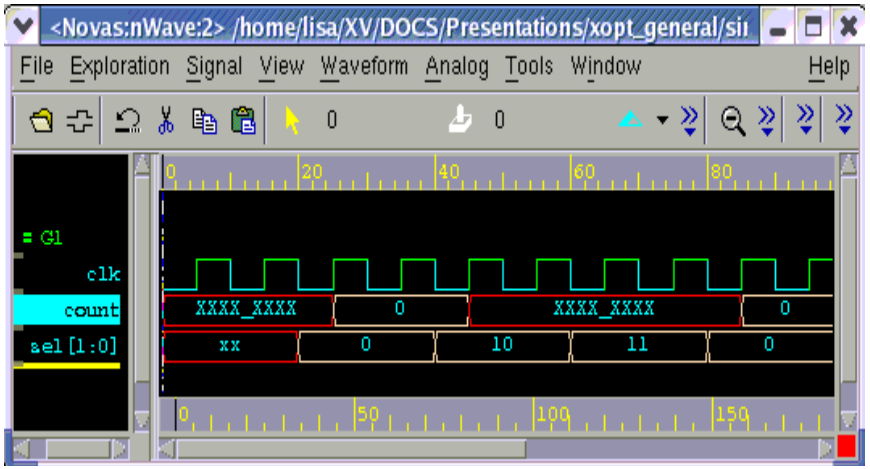
- Structural Analysis
- Simulation
- Modeling for X-Accuracy
- Formal Analysis

Structural Analysis and Simulation

- Structural analysis
  - Identifies potential issues
  - Can be very noisy
  - No sequential analysis

```
always @*
  case (state)
       00:   var1 = 2'b01;
       01:   var1 = 2'b11;
       10:   var1 = 2'bxx;
       11:   var1 = 2'b00;
  endcase
```

- Simulation-based
  - This is the existing infrastructure
  - Manual analysis of differences is slow and painful
  - Monitoring of X's is very noisy
  - Random initialization to eliminate X's can hide issues



Modeling for X-Accuracy

Trivial Example

```
reg f;
reg [1:0] g;
if (f===1'b0)
    g = 2'b00;
else if (f===1'bx)
    g = 2'b0x;
else
    g = 2'b01;
```

More Realistic, Yet Simple Example

```
reg [1:0] g, expr1, expr2, expr3;
always @(posedge clk)
begin
    if (f===1'b0)
       case (sel)
         1'b0:  g = expr1;
         1'b1:  g = expr2;
       endcase
    else if (f===1'bx)
        g = ???????????;
    else
        g = expr3;
end
```

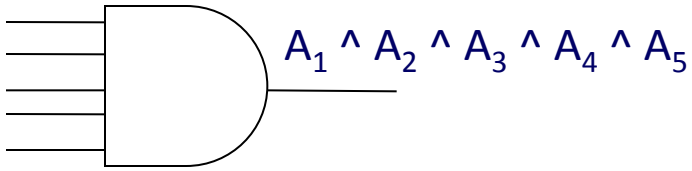
X-accurate Coding is Error Prone

Formal Analysis Point Solutions

- **Model checkers (ABV)** – tests all sequences of valid input scenarios (X's are tested both as a 1'b1 and a 1'b0)
  - Exhaustive analysis
  - VCD trace for failures
  - Capacity limitations exist
  - Must define functional behavior via assertions to prevent false fails

```
Define Functionality:
assume property  @(posedge clk)
      ( req  |=> ##[1:10] grant );
assert property @(posedge clk)
      ( grant && req  |=> !req );
```

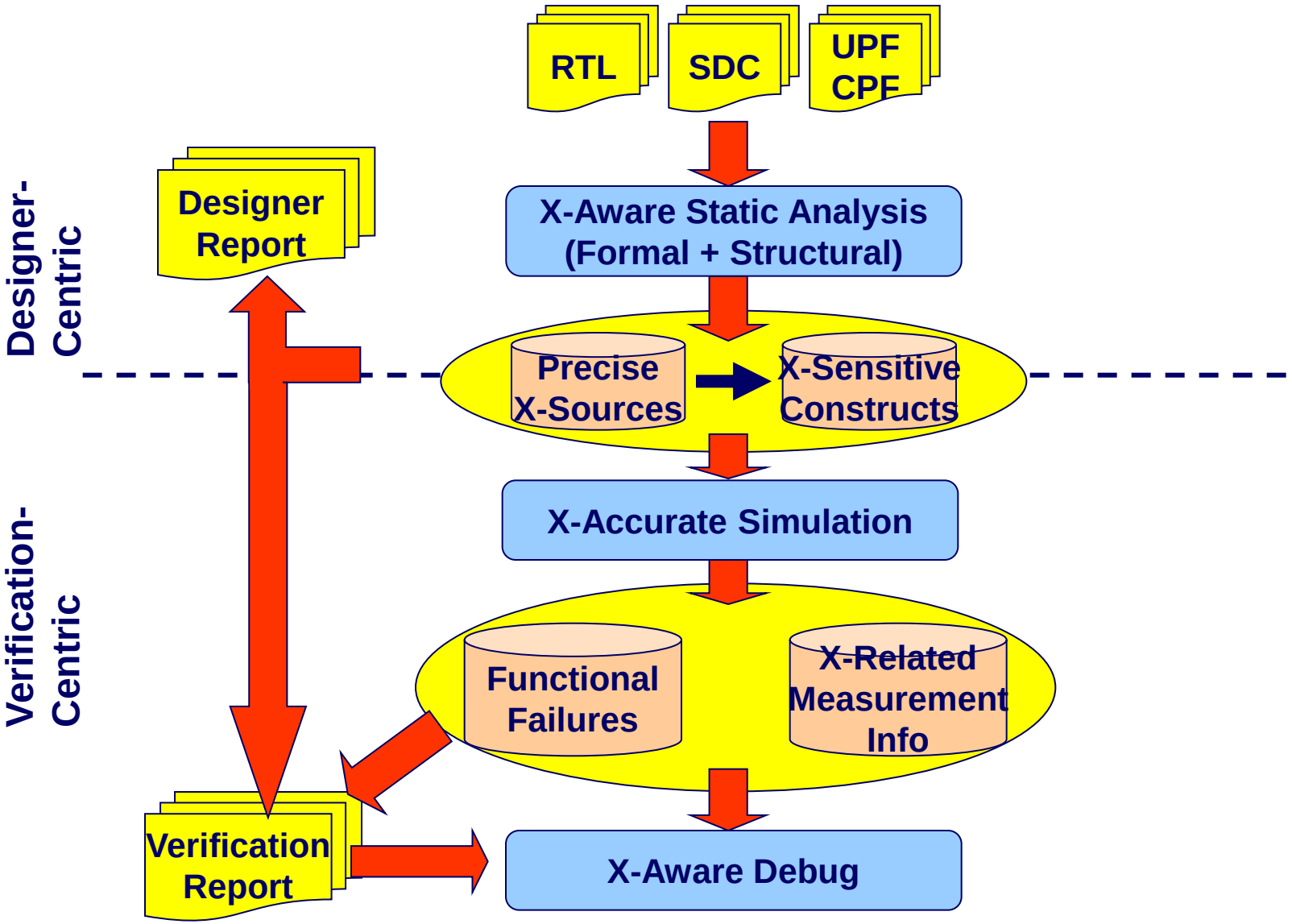
- **Symbolic simulators** – outputs are represented by a function of the input variables
  - Exhaustive analysis
  - VCD trace for failures
  - Capacity limitations exist
  - False failures



Proposed Solution

- Addresses both X-optimism and X-pessimism
- Combines both methodology and technology
- Brings to bear a combination of structural analysis, selective use of formal analysis, and simulation

Holistic Solution!



Reports

X-source Signals
- Type of X-source
- Links to Design Source

X-sensitive Signals
- Source Link
- Inputs to Which an X Can Propagate
- Debug Information

Simulator Messages
- Functional Errors Detected by Existing Checkers
- Time and Signal where X-optimism/X-pessimism Occurred

X-Net Statistics
- Counts of the Number of Transitions to X on all X-nets
- Provides test coverage information relative to X's

Holistic – addresses Optimism and Pessimism in the same way!

Simulation Debug Information

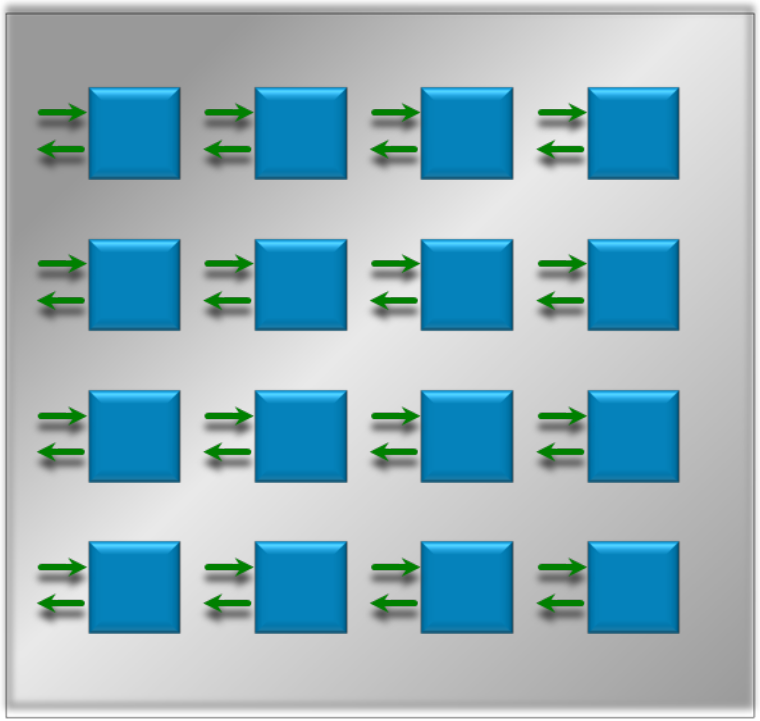
Simulation Messages

/**Running simulation with Real Intent Ascent XV generated X-optimism monitors ********/**
XV Info: X_OPT:out and X_OPT:in monitors are disabled at time 0 for instance `i2c_TOP
INFO: WISHBONE MASTER MODEL INSTANTIATED
status: 99500 done reset
XV Info: X_OPT:out and X_OPT:in monitors are enabled at time 100000 for instance `i2c_TOP
status: 109600 programmed registers
status: 113600 verified registers
status: 121600 generate 'start', write cmd 20
status: 11445600 write slave memory address 01
status: 21543600 write data a5
.
.
XV Warn: signal `i2c_TOP.cr[7:4] was sensitive to X-optimism at time 106049000

Statistics

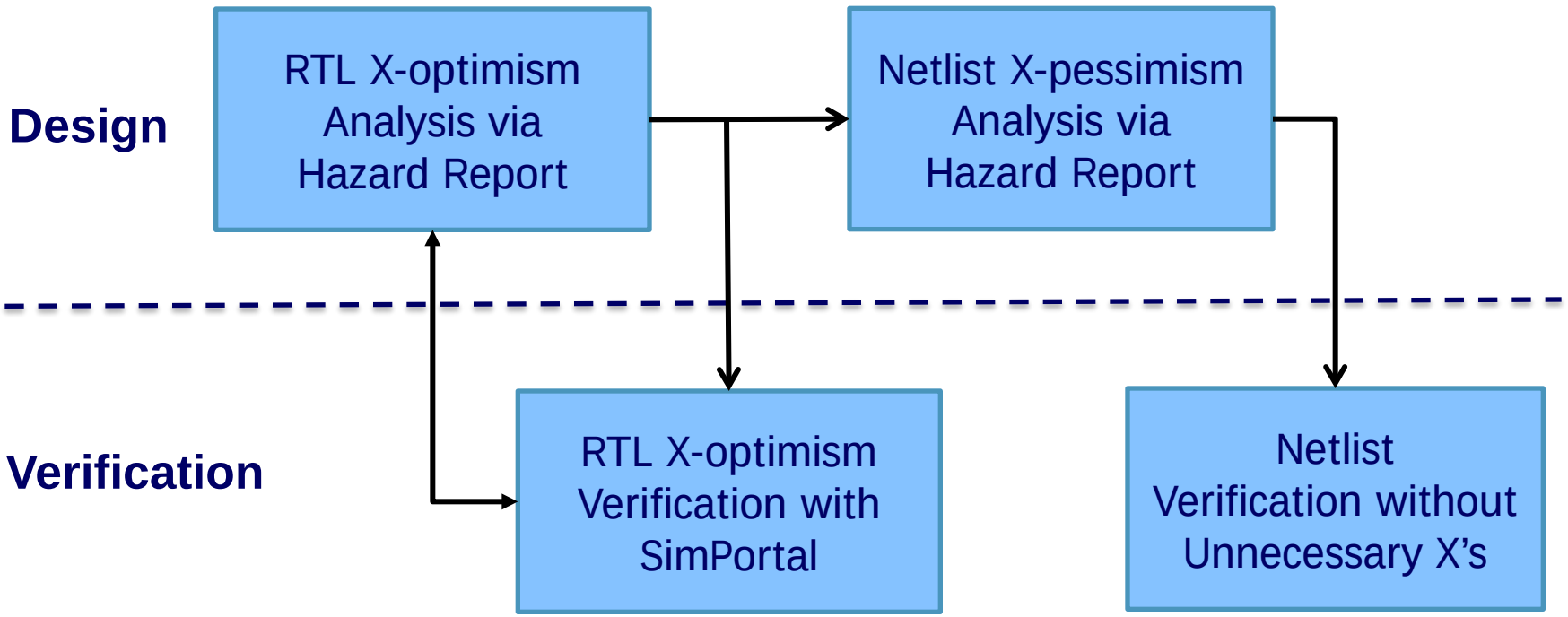
```
Control Input Signal Names
Matches   Signal Name
-------- ----- ------------
16        byte_ctrl_bit_ctrl_al
16        i2c_al
38492     wb_adr_i_2_0

Signals Sensitive to optimism
Matches    Signal Name
-------- ----- -------------
16        bit_ctrl.scl_oen
16        bit_ctrl.cmd_ack
16        bit_ctrl.c_state[17:0]
2         byte_ctrl.c_state[4:0]
16        byte_ctrl.shift
10        byte_ctrl.core_txd
2         byte_ctrl.core_cmd[3:0]
2         cr_7_4
38490     wb_dat_o_7_0
```

Hierarchical Approach Recommended



- Block-level X analysis
  - Easier debug.
  - Faster performance.
  - Fix issues.
- Full-chip analysis
  - Tuned X models applied to full chip.

X-Verification Flow



Solution Benefits

- Brings to bear a selective combination of structural analysis, simulation, and formal analysis
  - Addresses X-optimism at the RTL
  - Addresses X-pessimism at the Netlist
  - Isolates the offending X-source
- Holistic!
  - Addresses X-optimism and X-pessimism in the same way

Summary

- X's can mask functional bugs in the RTL and cause unnecessary X's in the netlist simulations.
- A solution must go beyond just identifying possible sources,
  - Detect and isolate real X-optimism bugs that can cause iterations/re-spins at the RTL level
  - Eliminate delays caused by X-pessimism in gate level simulations

Sign Off on Functionality at RTL	√
Minimize RTL-Netlist Iterations	√
Minimize Painful Debug	√
Sign Off with Confidence at Netlist	√