Is your Power Aware design really x-aware ?

Durgesh Prasad
Design and Verification Technology
Mentor Graphics
durgesh_prasad@mentor.com

Jitesh Bansal
Design and Verification Technology
Mentor Graphics
jitesh_bansal@mentor.com

***Abstract*— X-optimism is a precarious problem in RTL simulation. It can hide X bugs to cause serious issues in real silicon. Such hidden bugs are aggravated in power aware simulation due to injection of additional 'X' from power down regions. Traditional verification techniques such as tool generated assertions [4] and custom bind checkers [5] cannot catch such issues. Nowadays a new technique X-propagation is used to catch x-optimism related issues in RTL simulation. But this technique lacks the knowledge of power intent of design, causes unnecessary noise, and therefore not very useful in power aware simulation. In this paper, the authors would describe an effective technique to catch x related issues such as reset failures, wake up failures and x-optimism issues in power aware simulation.**

In this paper we present a method to use power aware knowledge on existing x-propagation technique for comprehensive x verification which is fully automated and provides ease of debug. Our solution will selectively apply X propagation technique according to system power state in a controlled way. This dynamic selection and controllability would ensure minimal noise, relevant x-propagation and better debug capability. The propagated x values can be observed in simulation waveforms and debug tools. This will catch x-optimism issues in a power aware simulation which are known to cause design failure at synthesis level.

Also, our solution will automatically insert SystemVerilog assertions to catch x-errors at the source. These assertions would be active according to current simstate of the system and they can also be used as an alternate for custom bind checker or low power assertion checks. This solution has advantage that it is fully automatic, comprehensive and free from user input. The downside is that it could generate some noise because the RTL morphing could be overly pessimistic.

In the paper we will further discuss the tradeoffs and methodology in detail.

Keywords—assertions; RTL; SystemVerilog; waveforms; debug tools; state elements; UPF; Power Domain; isolation; DUT;domain supply;simstate;X-optimism

I. INTRODUCTION

Power Aware verification has become increasingly critical for the semiconductor industry. Due to shrinking process geometry designers are focusing more on reducing static and dynamic power and it puts immense burdens on verification teams to ensure complete power aware verification.

The common trend is to start power aware verification once functional RTL verification is complete. The power aware behavior is imparted on the functional RTL design using UPF and liberty cells (optional). The various EDA vendors provide simulation tools to verify powers aware techniques like power shut-off, back-biasing and voltage scaling using UPF.

In simulation context these techniques are verified using corruption which means injecting x values on the signals of concern and propagating them further down the logic. These x-values require a competent handling, efficient tracking and painless debugging. In recent past there have been significant investments in this area and as a result good debug tools, faster power aware simulation and low-power assertion checks have been developed. Although these are good at catching x related issues visible at RTL level, they fail to catch certain silicon related issues arising due to x-optimism of RTL.

X-optimism [1] is when there is some uncertainty on an input to an expression or gate (the silicon value might be 0 or 1), but simulation comes up with a known result instead of X .For example SystemVerilog has an optimistic behavior when the control condition of an *if...else* statement is unknown.

```
always_comb begin
    if (sel) y = a;
    else y = b;//if sel is X (Potential mismatch with silicon)
end
```

Fig. 1 X-optimism example

There are techniques like xprop to catch the RTL x-optimism related issues but they are not well suited for power aware designs and need proper tuning and adjustments to be helpful for power aware design. Also these techniques cannot always point to the source of problem and it would be tedious to debug the actual bug. X-prop and X-trap technique customized according to power intent can be a good approach to this problem.

II. X SOURCES IN POWER AWARE DESIGN

A. RTL sources

There are several conditions where simulation will generate x logic. Some of which are as follows:

- Uninitialized 4-state variables
- Uninitialized state elements
- Unconnected module input ports
- User assigned x-values
- Bus contention
- Logic gates with 'x' inputs

B. Power Shut-down

The main idea of using power-shut down is to turn off massive unused parts of the design and, as a result, gain low current consumption. In UPF power-shut down corresponds to CORRUPT state of the power domain. In this power state, state elements powered by the domain supply and the logic nets driven by elements powered by the domain supply are corrupted. The simulation environment mimics the corruption by injecting X values. These X values can propagate further to other connected domains if isolation is not placed.

C. Back-biasing

These techniques are used to raise the threshold at which a transistor can change its state. While back bias slows the performance of the transistor it greatly reduces leakage. In UPF terms, the power domain has limited power, just enough to retain the state of its elements and cannot perform any switching activity.

In UPF [3] back-biasing corresponds to following state of the power domain:

- CORRUPT_ON_ACTIVITY

  In this simstate, the power characteristics of the primary supply set of power domain are insufficient to support activity in domain. Any activity would cause the corruption of the domain elements.

- CORRUPT_STATE_ON_ACTIVITY

  In this simstate, the power characteristics of the primary supply set of power domain are insufficient to support activity inside state elements, whether that activity would result in any state change or not. Any activity on the input of the state element would cause its corruption.

- CORRUPT_STATE_ON_CHANGE

  In this simstate, the power characteristics of the supply set are insufficient to support a change of state for state elements. Any activity that results in state change for state element would cause the corruption of that state element.

D. Voltage Scaling

In this technique, two interacting power domains operate with different voltage ranges. In this case, logic 1 value might be represented in the driving domain using a voltage that would not be seen as an unambiguous 1 in the receiving domain. Level-shifters are inserted at a domain boundary to translate from a lower to a higher voltage range, and sometimes from a higher to a lower voltage range as well. The translation ensures the logic value sent by the driving logic in one domain is correctly received by the receiving logic in the other domain. In absence of these level shifters, the signals get corrupted while crossing the domain.

E. Power aware cells

Power aware verification makes use of some special cells which can generate 'x' values when their power is off or their control signals are 'x'. These cells can be divided into following categories.

1) Isolation cell: It is a design element that passes logic value during normal mode and clamps its output to some specified logic value when control signal is asserted. If the control signal is x or the power supply of the cell is off then it drives 'x' value at output.

2) Level-shifter cell: It is a design element that translates signal values from an input voltage swing to a different output voltage. If the power supply of the cell is off then it drives 'x' value at output.

3) Retention cell: These cells are used to impart retention behavior on state elements. If the power supply of the cell is off or due to some incorrect protocol it can drive 'x' value at the retention register output.

4) Buffer cell: Buffers can be inserted to power the ports of a hard macro. These cells drive 'x' values if their power goes off.

III. EXISTING SOLUTIONS FOR POWER RELATED X ISSUES

Some of the existing solutions used to find X related issues in power aware design are:

A. Low power checks and custom bind checker

Today, Most of the EDA vendors provide automated assertions to verify the various UPF protocols such as Control Signal corruption check, Missing isolation check, power control sequence protocol check etc.[4] A lot of assertions are inserted in RTL to catch the power intent bugs. Designers also specify their own checks using UPF command bind_checker [5]. These assertions are able to identify many power intent issues but they are failed to catch X-optimism issues. Also, it is painstaking task for user to specify bind_checker commands for complete design and there is always a chance to miss some protocol.

B. X-propagation and X-trapping

X-propagation is the latest technique to address X-optimism by propagating X values forward in time. It mostly involves if statements, case statements, and conditional assignments. For example, when a conditional expression has the value of X, the X-propagation enhanced simulator changes the language semantics to propagate X values. These values

can be observed in simulation waveforms and the downstream logic is affected by the propagated X values. If the Xs are not blocked or handled correctly in the design, the simulation could fail. In particular, the design's silicon implementation could subject to similar random failures, which is completely missed by normal RTL simulator.

X-trapping is a debugging technique used alongside X-propagation to efficiently debug the 'X'. When an X appears in simulation, it is a daunting task to triage and trace its cause. It is better to detect the X at the moment it occurs. Various tools implicitly instrument System Verilog (SVA) assertions [1][2] to do this. During simulation, these assertions trap Xs at their sources. They are managed with the current assertion infrastructure—just like other assertions in the design.

X-prop and X-trap techniques modified the existing RTL in following way:

TABLE I. X-PROP AND X-TRAP EXAMPLE

RTL	X-PROP AND X-TRAP

always_comb
begin
if(sel)
y1 = a;
else
y1 = b;
end
	always_comb
	begin
	if($isunknown(sel)) begin
	assert (!$isunknown(sel))
	else $error("sel goes X");
	y1 = x;
	end else if(sel)
	y1 = a;
	else
	y1 = b;
	end

However, in power aware, this technique cannot be used in its existing form because the power down region generates lot of Xs which are not error scenarios and this technique will cause lot of false assertion failures, making it difficult to identify actual X issue. Also the issues which are caused due to corruption (Un-Initialization) of various power control signals would not be caught at the source by the existing technique.

IV. MOTIVATION BEHIND THE PAPER

In recent past Power Aware verification at RTL level has grown tremendously and several papers have been written on X-verification technique using powerful assertions and bind checkers. EDA vendors also provide automated assertions and dedicated power aware debug to ease the verification. In spite of these tools and techniques, there is a real risk of either not catching subtle low-power bugs or caching it very late during expensive gate-level simulation, or not at all, causing disastrous functional product failures.

The motivation behind this paper is to demonstrate the technique proposed by us to

1) Catch some of these issues at RTL level itself by combining x-prop technique with power aware intent.
2) Provide clean and efficient debugging of source of the problem using x-trap in conjunction with power intent.
3) Provide efficient automated checks to catch various power related issues.

V. PROPOSED XPROP-PA SOLUTION

We propose to take the X-propagation technique one step further and combine it with power intent of design to provide a complete and efficient solution.

A. *Catch x-optimisim related issues in Power Aware Simulation*

Let's consider a simple power aware example in which signal y1 crosses from power domain PD1 to power domain PD2.

TABLE II. POWER AWARE EXAMPLE

PD1	PD2

assign sel = in1 & in2;
 	
always_comb
begin
if(sel)
y1 = a;
else
y1 = b;
end
	always_comb
	begin
	if(y1)
	y2 = a;
	else
	y2 = b;
	end

Following table summarizes the concerned state of PD1, PD2 and various outputs.

TABLE III. OUTPUTS WHEN PD1 GOES OFF

PD1	PD2	sel	y1	y2
Off	On	x	x	b

The desirous value of y2 is 'x' but a normal power aware simulation will impart it value 'b'. This x-optimism issue can be handled by applying xprop in PD2.

```
always_comb
  begin
      if($isunknown(y1)) begin
          assert (!$isunknown(y1)) else
          $error("y1 goes X");
          y2 = x;
      end else if(y1)
          y2 = a;
       else
          y2 = b;
  end
```

Fig. 2 Resultant RTL after applying xprop

Following table summarizes the xprop results.

TABLE IV. OUTPUT AFTER APPLYING X-PROP

PD1	PD2	sel	y1	y2
Off	On	x	x	x

B. Provide a noise free efficient x-verification

Noise: Excessive useless error messages which do not point to actual design issue instead interfere in debugging the actual design problems.

1) Automated noise reduction

Consider the same example mentioned in Table II, after applying xprop on power domain PD1 and PD2 following outcome would be achieved.

TABLE V. XPROP SIMULATION RESULTS

PD1	PD2	sel	y1	y2	Assertion Failure of 'sel'	Assertion Failure of 'y1'
On	On	x	x	x	Yes	Yes
Off	On	x	x	x	Yes	Yes
Off	Off	x	x	x	Yes	Yes
On	Off	-	-	x	-	Yes

In power aware simulation, X value on 'sel' is expected when PD1 goes OFF. So, any xprop assertion failure in this power domain is a noise. Similar is the case with 'y1' when PD2 goes OFF. There can be lot of such dummy failures when power domain goes off and they can interfere in identifying actual x-optimism issue (just like the one described in Table III). Even a single lost bug because of noise can have disastrous effects.

We propose to reduce the noise by controlling the xprop behavior according to the current simstate of the power domain. When power goes OFF the xprop would be disabled for that domain, thus reducing the dummy errors from that domain.

The instrumented behavior of PD1 logic would look like Fig. 3.

```
assign control = function(PD1 current  simstate);

always_comb
  begin
      if($isunknown(sel) && control) begin
          assert (!$isunknown(sel)) else
          $error("sel goes X");
          y1 = x;
      end else if(sel)
          y1 = a;
       else
          y1 = b;
  end
```

Fig. 3 Resultant RTL after applying Xprop-PA

Similar behavior will be imparted on logic of PD2 where xprop behavior would be governed by current simstate of PD2.

The proposed behavior has following benefits:

- Noise will be reduced.
- Need to debug only actual x-issues.
- Fewer chances of lost bugs in noise.
- Assertion failure reduction would cause simulation performance boost.

The final output looks as per following table:

TABLE VI. XPROP-PA SIMULATION RESULTS

PD1	PD2	sel	y1	y2	Assertion Failure of 'sel'	Assertion Failure of 'y1'
On	On	x	x	x	Yes	Yes
Off	On	x	x	x	No	Yes
Off	Off	x	x	x	No	No
On	Off	-	-	x	-	No

2) Handling of UPF simstates

Xprop behavior is controlled according to current simstate of primary supply of power domain. We are proposing following behavior for each simstate:

- CORRUPT
  In this simstate, all the state and logic elements would be corrupted. So it is recommended to switch off xprop for this state to avoid noise.
- CORRUPT_ON_ACTIVITY (COA)
  In this simstate, any activity on the input (including X) would trigger corruption of connected logic by power aware simulator. So it is advised to switch off xprop in this state too.
- CORRUPT_STATE_ON_ACTIVITY (CSOA)

  In this simstate, power aware simulator would corrupt state elements whenever there is any activity on any input of state element, while combinatorial logic would still active. So, xprop should be enabled for combinatorial logic only.
- CORRUPT_STATE_ON_CHANGE (CSOC)
  In this simstate, power aware simulator would corrupt only when output of state element changes. Changed output can impact other state or combinatorial element which power aware simulator cannot catch. So, we propose to turn on xprop on state as well as combinatorial elements.

Here is the summary of our proposed xprop behavior in various UPF simstates for combinatorial and state elements.

TABLE VII. XPROP-PA CONTROL LOGIC FOR UPF SIMSTATES

Simstate	Xprop for Combinatorial logic	Xprop for State elements
CORRUPT	OFF	OFF
COA	OFF	OFF
CSOA	ON	OFF
CSOC	ON	ON

3) Behavioral Models

There are some models of PLL, memories and analog parts that designer prefer to exclude from being corrupted in power aware simulation. Power aware vendors provide ways to exclude such blocks for power aware processing. X-propagation is also meant to find RTL-GLS mismatches for synthesizable code only. So it is recommended to exclude the behavioral models in Xprop-PA too. It would also make the simulation faster and less noisy.

4) User controlled X-propagation

In many power aware designs, design reset happens after power is ON, commonly known as Power on Reset (POR). During the period between power ON and reset, the design is in unknown state with many signals have X values.

Fig. 4 wave diagram showing POR

These X's are noise and need to be filtered from xprop. We recommend that x-prop should be enabled only after reset is performed. Either it could be an automated reset detector or a manual input from user to control the Xprop behavior. This solution would further reduce the noise to some extent.

5) Controlled Assertion Failures

There should be mechanism to limit the fail count of any assertion. Sometimes same assertion is reported large number of times at each clock activity, creating noise. Tool should provide default as well as user control input to limit the failure count.

C. Efficient debugging

1) SV Assertions

SV Assertions play very important role to catch the bugs in RTL verification and Xprop-PA can use them to trap the 'X' at its source. This technique is known as X-trapping [2]. It pin points to exact bug and user can easily debug it with tool assertion window just like any other RTL assertion.

2) Design element categorization

The tool debug capability can be further enhanced by grouping the assertion messages according to type of RTL construct. The label of error message should be created in such a way that user can easily identify the construct for which error is flagged. In this way, user can give preference to more critical constructs.

We are suggesting following labels for some of the common constructs:

- XPROP-FF
  This label should be used when any asynchronous control of a RTL flop goes x.

  *# ** Error: XPROP-FF: 'reset' goes X.*
  # Time: 2 ns Scope: tb.dut File: ./src/vl_file/lib/rtl.v Line: 6

- XPROP-CLK
  This label should be used for clocks.

  *# ** Error: XPROP-CLK: 'clk' goes X.*
  # Time: 12 ns Scope: tb.dut File: ./src/vl_file/lib/rtl.v Line: 8

- XPROP-LATCH
  This label is for latch enable of a RTL latch.
- XPROP-FSM
  If state variable of a FSM goes x then this label should be used.

  *# ** Error: XPROP-FSM: 'cst' goes X.*
  # Time: 12 ns Scope: tb.dut File: ./src/vl_file/lib/rtl.v Line:28

- XPROP-MEM
  This label is for memories when read enable, write enable or address bus of a memory goes x,
- XPROP-COMB
  This is generic label for all the combinatorial logic. It can be further divided according to logic.

  *# ** Error: XPROP-COMB: 'ack' goes X.*
  # Time: 10 ns Scope: tb.dut File: ./src/vl_file/lib/rtl.v Line: 46

D. Automated power aware checks

In addition to trapping 'X', assertions can also be used in validating many UPF protocols. Some of the useful checks being done by our proposed solutions are as follows:

1) Control signal corruption check

Power aware cells inserted by UPF are governed by control signals coming from HDL. These signals are not actually connected to DUT but driven at test bench for power aware verification only. A functional xprop would not be able to catch 'x' issues on these signals as they are not used in any DUT logic in non-power aware simulation. If any of these signals goes to 'X' then power aware simulation fails but it

could be painful to find the source of the problem. Our Xprop-PA solution would trigger the assertion at its source.

One such example is when save/restore signal of retention cell is 'X' at power up. The state logic controlled by these save/restore signals would trigger xprop assertion.

Fig. 5 Wave diagram showing ret signal is x at power up

*# ** Error: XPROP-FF: 'ret' goes X.*
 # Time: 2 ns Scope: tb File: ./src/vl_file/lib/dut.v Line: 37

Another example is the power up failure of the power domain. This happens when control signal of the switch controlling the primary supply of the domain is coming through a power down region. In this case xprop would flag the assertion for the power switch as soon as the control signal goes x.

2) Missing isolation

It is one of the most common scenarios of power aware design failure when power domain boundaries are not properly isolated, and many such issues are missed in RTL verification because of the X-optimism. Example referred in Table II is the case of missing isolation cell at domain boundary PD1-PD2.

3) Reset Failures

It is very common X-propagation issue, when 'X' value on asynchronous controls does not affect RTL simulation because of X-optimism.

```
always @(posedge clk_s or negedge rst_n_s)
begin: ff_verilog_model
    if(rst_n_s == 1'b0)
        q_s <= 1'b0;
    else
        q_s <= d_s;
end
```

Fig. 6 DFF with asynchronous reset

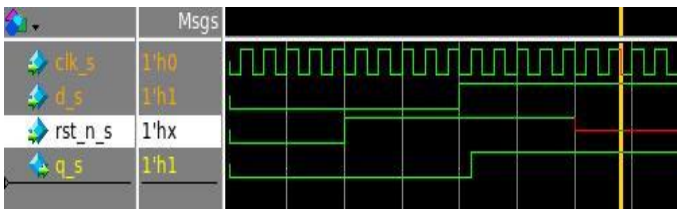

Fig. 7 Wave diagram showing DFF output when reset goes x

From RTL simulation, DFF with async-reset is not sensitive to 'x' at the reset pin which is a potential bug. It would be caught at its source by Xprop-PA.

VI. RESULTS

The ideas proposed in the paper are the outcome of our experiment on various designs. Some of the proposed methods like "controlled assertion failure" and "design element categorization" are the based on the user feedback from non-power aware customer designs. Other ideas like "noise reduction" and "power aware checks" have come out of our research on power aware designs. The simulation performance and debug features of the proposed solution are good enough to be used for active development.

VII. CONCLUSION

In this paper we have discussed how xprop can be used for power aware designs in a controlled way to catch x related issues.

Firstly, we started with explaining the problem of noise in xprop simulation on power aware designs. This is primarily due to corruption induced as a result of various power saving techniques e.g. power shut-off; back biasing, voltage scaling and power cells.

Secondly, we proposed a controlled xprop mechanism based on the current simstate of the power domain to reduce noise and catch x-optimism related issues. We also proposed some customization in the tool to handle POR and other noise related issues.

Then, we suggested some of the debugging techniques for efficient debugging.

Finally we explained some of the power aware automated checks performed by xprop and how they can be useful in catching x issues at the source rather than doing a trace back and debug.

We have good confidence that the proposed power aware xprop solution would be a strong and efficient tool to uncover power related x bugs and reduce gate level simulation effort.

VIII. ACKNOWLEDGEMNT

We would like to express our great thanks to Yeung Ping who helped us a lot in early stages of our xprop solution. Many thanks to every member of Mentor Power Aware team who gave us several enlightening ideas and suggestions for this paper. Finally, thanks to all our xprop customers, whose feedback gave us confidence to write this paper.

IX. REFERENCES

[1] Stuart Sutherland, "I'm Still In Love With My X!", DVCon 2013

[2] Don Mills, "Being Assertive With Your X", User2user 2013

[3] UPF LRM IEEE_1801_2009, "IEEE standard for Design and Verification of Low Power Integrated Circuit".

[4] Bembaron F., Kakkar S., Mukherjee R. and Srivastava A., "Low Power Verification Methodology using UPF", DVCon 2009

[5] Harsh Chilwal, Manish Jain, Bhaskar Pal, "An Integrated Framework for Power Aware Verification"

[6] Colin Dong,Cherin Joseph, "Catch hidden x bugs in RTL simulation with X-propagation technology",SNUG 2012