Is your Power Aware design really x-aware?

Durgesh Prasad, Design and Verification Technology
Jitesh Bansal, Design and Verification Technology
Introduction

• Power Aware simulation
  – Inject x-values to mimic design shut-off, back biasing, voltage scaling, PA cells shut-off.
  – Simulation can catch many design issues arising due to these power techniques.
    • Automated power aware checks and assertions (bind checker) further help in catching these issues.
  – Issues arising due to x-optimism/pessimism of RTL can not be caught by a raw power aware simulation.
X Optimism: Condition stmts

- Appears commonly in simulation
  - If-statements:
    
    ```
    if (cond)  
        reg_a <= f0;
    else       
        reg_a <= f1;
    ```
  
  - Case-statements:
    ```
    case (cond)
        2'b00:   reg_b <= f0;
        2'b11:   reg_b <= f1;
        default: reg_b <= f2;
    endcase
    ```

- h/w will consider all scenarios

- "Simulation" is not simulating what happens in actual h/w
  - Some bugs cannot be detected using simulation
X-optimism solution: X-Prop

- Propagating X values forward

```c
if (cond)
    reg_a <= f0;
else
    reg_a <= f1;
```

### Results from synthesis are unpredictable.

Simulation may not exercise the worst scenario.

-> Propagating Xs
X-Optimism Solution: x-Trap

- Conditional statement

```verilog
always_comb begin
    if (cond)
        reg_a <= f0;
    else
        reg_a <= f1;
end
```

- Conditional statement with implicit assertion

```verilog
always_comb begin
    assert !$isunknown(cond);
    if (cond)
        reg_a <= f0;
    else
        reg_a <= f1;
end
```

To catch Xs in condition expressions ASAP
X-Prop PA solution

- Catch x-optimism issues

<table>
<thead>
<tr>
<th>PD1</th>
<th>PD2</th>
<th>sel</th>
<th>y1</th>
<th>RTL sim (y2)</th>
<th>X-prop PA (y2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Off</td>
<td>On</td>
<td>x</td>
<td>x</td>
<td>b</td>
<td>x</td>
</tr>
</tbody>
</table>
X-Prop PA (Handling Noise)

- **Automated Noise Reduction**

<table>
<thead>
<tr>
<th>PD1</th>
<th>PD2</th>
<th>sel</th>
<th>y1</th>
<th>y2</th>
<th>Error for ‘sel’</th>
<th>Error for ‘y1’</th>
<th>Xprop-PA Error for ‘sel’</th>
<th>Xprop-PA Error for ‘y1’</th>
</tr>
</thead>
<tbody>
<tr>
<td>On</td>
<td>On</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Off</td>
<td>On</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Off</td>
<td>Off</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>On</td>
<td>Off</td>
<td>-</td>
<td>-</td>
<td>x</td>
<td>-</td>
<td>Yes</td>
<td>-</td>
<td>No</td>
</tr>
</tbody>
</table>
**X-Prop PA (Handling Simstates)**

- `xprop_pa_logic` can be defined by the following table:

<table>
<thead>
<tr>
<th>Simstate</th>
<th>Xprop for Combinatorial logic</th>
<th>Xprop for State elements</th>
</tr>
</thead>
<tbody>
<tr>
<td>CORRUPT</td>
<td>OFF</td>
<td>OFF</td>
</tr>
<tr>
<td>CORRUPT_ON_ACTIVITY</td>
<td>OFF</td>
<td>OFF</td>
</tr>
<tr>
<td>CORRUPT_STATE_ON_ACTIVITY</td>
<td>ON</td>
<td>OFF</td>
</tr>
<tr>
<td>CORRUPT_STATE_ON_CHANGE</td>
<td>ON</td>
<td>ON</td>
</tr>
</tbody>
</table>
The proposed solution uses SV Assertions, which is known for its controllability and ease of debug.

Design element categorization

- XPROP-FF
  
  # ** Error: XPROP-FF: 'reset' goes X.
  

- XPROP-CLK
- XPROP-LATCH
- XPROP-FSM
- XPROP-COMB

User controlled x-propagation

- Provide enabling and disabling of x-prop logic based on timing to handle POR.
X-Prop PA(Automated checks)

- Control signal corruption check
  - save/restore signal(ret) of retention cell is ‘X’ at power up(a potential bug). The state logic controlled by these save/restore signals would trigger xprop assertion.

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

  – From RTL simulation, DFF with async-reset is not sensitive to ‘x’ at the reset pin which is a potential bug (would be caught by x-prop assertion)

```verilog
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
```
Conclusion and References

• Conclusion
  – A controlled x-prop PA solution can catch x-optimism related issues specific to power aware designs.
  – The various simstates of the power domain can be simulated to catch potential issue without generating noise.
  – Techniques like “controlled assertion failure” and “design element categorization” can make debugging user friendly.
  – This technique also provide automated checks like “control signal corruption” and “reset failure”.

• References
  – Stuart Sutherland, “I’m Still In Love With My X!”, DVCon 2013
  – Mike Turpin, “The Dangers of Living with an X” ARM 2003