Complete verification of SoC initialization is a fundamental requirement and a prerequisite to ensure successful chip operation. At MediaTek we break this down into two components:
1) Global Reset Verification
2) X-State Detection Verification
At MediaTek we have setup an automated flow for verifying these initialization features using Formal techniques.

**Global Reset Sources**

SoC initialization complexly utilizes many sources:
- Power-on Reset
- Hardware Reset
- Software Reset
- Interrupt Driven Reset
- Watchdog Timer Reset

**X-State Sources and X-Optimism**

X-State in SoC designs can come from many sources:
- Primary: Uninitialized Registers
- Primary: X assignments in RTL
- Secondary: Bugs in the design due to RTL coding
  - Example: Out of Bound Array Selecting Generates an X
    
    \[
    \text{reg} [9:0] \text{ w_slot}; \\
    \text{assign } \text{ y_data } = \text{ w_slot[x_select[3:0]]}; \\
    \]

  X-Optimism in RTL simulations causes potential bugs to be missed:
  - Typically affects if/else and case logic in designs
  - Most often cause bugs to be missed during initialization

Resettable Flip-Flop must not output an X-State after the initialization sequence is completed:
- X-States in a design are acceptable as long as the design functions properly
- Uninitialized registers and other X sources should not overwrite a Flip-Flop that has been initialized with a reset

**Global Reset Assertion**

Users must verify that all resets reach their intended targets with correct polarities.

**X-State Detection Assertion**

Users must verify that all Flip-Flops that have been reset aren’t overwritten by an X value after initialization:

- All types of reset can be verified (typically asynchronous reset)

**Example: Generated SVA Assertion**

\[
\text{x_check_id0: assert property (@(posedge clk) } \\
\text{ ^top.u1.my_reg !== 1'bX );} \\
\]

These assertions are created for all resettable Flip-Flops in the design and the X-State detection is verified by Questa Formal.

**Formal Verification Flow**

Following is a summary of the results for the two types of initialization verification that was performed. The tool used in generating the SVA assertions and formally verifying the results was Questa Formal from Mentor Graphics.

<table>
<thead>
<tr>
<th>Category</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Design Size</td>
<td>3,738,047 register bits</td>
</tr>
<tr>
<td>SVA</td>
<td>14,835 assertions</td>
</tr>
<tr>
<td>Run Time</td>
<td>gen_sva: 61 min</td>
</tr>
<tr>
<td></td>
<td>Compile: 6.5 hr</td>
</tr>
<tr>
<td></td>
<td>Prove: 2.3 hr</td>
</tr>
<tr>
<td>Formal Result</td>
<td>Fired: 462</td>
</tr>
<tr>
<td></td>
<td>Proven: 14,373</td>
</tr>
<tr>
<td></td>
<td>Inconclusive: 0</td>
</tr>
<tr>
<td>Bug</td>
<td>248 reset connection errors from 3 modules</td>
</tr>
</tbody>
</table>

**Formal Verification Results**

**Conclusion**

- Highly automated Formal techniques avoid the large amount of manual effort required by simulation-based verification to verify global reset schemes and X-state detection
- Formal techniques are completed within hours without any inconclusive results
- Other automated applications for Formal techniques will be explored to reduce manual effort and achieve higher verification confidence