Reset and Initialization: the Good, the Bad and the Ugly

Ping Yeung, Mentor Graphics
ping_yeung@mentor.com

Kaowen Liu, MediaTek Inc.
kaowen.liu@mediatek.com
Agenda

• Initialization Challenge
• The Register Status
  – 5 scenarios of register corruption
  – Assertions for registers
• The Initialization Sequence
  – The 3 phases
• Verification Methodology
  – Results
  – 5 corruption cases
• Summary and Reference
Initialization Challenge

• One daunting challenge of developing a low-power SoC design is how to fully verify its power-up, reset and initialization sequences.

• All of these possible reset sources are combined together to initialize the design
  – Power-on reset
  – Hardware reset
  – Software reset
  – Interrupt reset
  – Watchdog timer reset
The Register Status

• To understand the internal status of a design, our methodology classifies design registers into:

  – **GOOD** registers - are initialized properly,

  – **BAD** registers - are not initialized

  – **UGLY** registers - are initialized, but are subsequently corrupted
The Example

- RegC is a FSM state register whose input function logic contains the outputs of RegA and RegB

The FSM, RegC, should not be corrupted
The Scenario 1

- Reset Propagation Problem
  - reset signals failed to reach RegA and RegC correctly

Same issue when clock signals are Xs
The Scenario 2

- Reset Timing Problem
  - reset signals to RegA and RegC did not meet protocol

RegA is not initialized correctly; RegC is corrupted by RegA.
The Scenario 3

• X-propagation Problem:
  – reset signals assert correctly; RegB is not initialized correctly; RegC is corrupted by RegB.
The Scenario 4

• Power Sequence and Isolation Problem:
  – RegB is powered down; driving X; and not isolated
The Scenario 5

- Clock Gating Problem:
  - RegB is controlled by gated clock that is turned off

Clock to RegB is gated off; RegC is corrupted by RegB.
Assertions for Registers

- Ensure good clocks and resets
- Catch corruption with formal verification

Some assertions are available implicitly with X-prop simulation
The Initialization Sequence

• 3 Phases in Initialization Sequence

**Power on Phase:**
Reset On
No Clock

- Bad
- Good
The Initialization Sequence

- 3 Phases in Initialization Sequence

**Power on Phase:**
- Reset On
- No Clock

**Reset Propagation:**
- Reset On
- Clocking

- **Bad**
  - Corrupted by data-in
  - Synchronous reset
    - Cleared by data-in
  - Bad

- **Good**
  - Ugly
The Initialization Sequence

• 3 Phases in Initialization Sequence

Power on Phase:
Reset On
No Clock

Good

Bad

Reset Propagation:
Reset On
Clocking

Good

Bad

Synchronous reset
Cleared by data-in

Corrupted by data-in

Functional Phase:
Reset Off
Clocking

Good

Bad

Cleared by data-in

Corrupted by data-in

Ugly

Good

Ugly

Corrupted by data-in
Verification Methodology

• Reset & Clock Trees
  – Static Checks on reset trees and clock trees
  – Formal connectivity check: verify control and distribution of reset and clock signals

• Initialization
  – X-prop simulation: monitor X propagation and usage
  – Formal initialization check: find true un-initialized registers; avoid X pessimism

• Initial Status
  – Formal modeling checking: ensure good registers are not corrupted; turned ugly
## Results

<table>
<thead>
<tr>
<th>Design Complexity</th>
<th>Design 1</th>
<th>Design 2</th>
<th>Design 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of register bits</td>
<td>305</td>
<td>47016</td>
<td>43622</td>
</tr>
<tr>
<td>Number of latch bits</td>
<td>0</td>
<td>592</td>
<td>0</td>
</tr>
<tr>
<td>Number of RAMs</td>
<td>2</td>
<td>0</td>
<td>64</td>
</tr>
<tr>
<td>Number of asynchronous resets</td>
<td>3</td>
<td>13</td>
<td>16</td>
</tr>
<tr>
<td>Number of synchronous resets</td>
<td>2</td>
<td>33</td>
<td>35</td>
</tr>
<tr>
<td>Number of clocks</td>
<td>3</td>
<td>5</td>
<td>12</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Register Status information</th>
<th>Design 1</th>
<th>Design 2</th>
<th>Design 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Good registers</td>
<td>38%</td>
<td>50%</td>
<td>34%</td>
</tr>
<tr>
<td>Bad registers</td>
<td>58%</td>
<td>9%</td>
<td>66%</td>
</tr>
<tr>
<td>Ugly registers</td>
<td>4%</td>
<td>41%</td>
<td>&lt;1%</td>
</tr>
</tbody>
</table>
Corruption Case #1

- Reset signals passed from one module to another
  - Reset signals are unknown when via module is powered down

![Diagram showing reset signal flow between power domains](image)
Corruption Case #2

• Reset control signals from another module
  – Foreign reset control signals may disable initialization after power cycle
A good register was corrupted by an X source (an X assignment inside the default case branch).
A good (synchronous reset) register was corrupted by a group of uninitialized bad registers. The TX registers were not enabled and were holding their un-initialized values.
Corruption Case #5

- Questa Formal was used
- Targeted FIFO to ensure it functions correctly
As SoC designs get complex, reset and initialization become harder to verify

We discussed:
- The register status: Good, Bad and Ugly
- Assertions for registers
- The initialization sequence
- Verification Methodology
- Results
Reference

DVCon 2015

– Multi-Domain Verification: When Clock, Power and Reset Domains Collide
  • Ping Yeung, Erich Marschner, Mentor Graphics
  • Kaowen Liu, MediaTek Inc.

– Addressing the Challenges of Reset Verification in SoC Designs
  • Chris Kwok, Priya Viswanathan, Ping Yeung, Mentor Graphics