Exhaustive Reset Verification
Enablement: Client PCIE Design Reset Verification
Case Study

Rohit Kumar Sinha, Intel India
Praveen Dornala, Intel India
Motivation

• With the increased complexity of Intel Client SoC design,
  – the hierarchical reset architecture has become very complex
  – increase in reset signaling complexity with the emergence of multiple reset domains
  – create new verification challenges that aren’t addressable by RTL simulations.

• Because of the different flavors of IPs
  – independent “reset domains” can be created by complex reset sequences or reset structures
  – metastability and reconvergence issues similar to the failures seen in asynchronous CDC.
  – tangible need to provide automated, exhaustive structural and functional reset signaling checks.
Exhaustive Reset Verification

1. Reset Tree Evaluation
   • Power On Reset, Watchdog Timeout reset
   • Debug reset, Software reset, and Loss of Clock reset.
   • Construction Issues

2. Reset Domain Crossing
   • Application of resets- Sync/Async Behaviour
   • Same reset used as both sync/async in a single module leading to synth vs sim mismatches
   • Combinational Logic in Reset Path

3. Reset Convergence & Reconvergence
   • Reset splitting and converging
   • Reset bus synchronizers reconverging

If rst1 is asserted while rst2 is not asserted, the asynchronous data fromdff1 will metastability on dff2
Evidence: Additional Metastability Risk Uncovered

1. RESET combo glitch: This could lead to a glitch in the reset path due to combo logic
Evidence: Additional Matastability Risk Uncovered

2. RESET convergence Issue appears when reset get split and converge
Evidence: Additional Matastability Risk Uncovered

3. RESET is always active HIGH
Evidence: Additional Matability Risk Uncovered

4. Asynchronous wrongly connected to data without synchronizer
Reset Domain Crossing

Introduction using a simple Transmit and Receive Flop

 ✓ Consider 2 flops Tx and Rx on the different and same clock domain.

 ✓ All resets have to assert 0 to clear the flops.

 ✓ Tx flop is on reset (R1)
 ✓ Rx flop is on reset (R2)

 ✓ Reset (R1) asserts when R2 is de-asserted
Reset Domain Crossing

Concept

Reset Domain Crossing is the phenomena when two logic blocks (or flops) are reset using two different asynchronous resets. When we are going into reset, the first reset can assert and make the first block (or first flop) to change asynchronously. This change can make the second block (or the receiver flop) to go meta-stable. This meta-stability caused due to resets is the problem that we are trying to avoid. The cause of this kind of meta-stability is what is known as reset domain crossing.
Reset Assertion Ordering

```
<table>
<thead>
<tr>
<th>Reset Signal</th>
<th>Source</th>
<th>Destination</th>
<th>Clock</th>
<th>FLR, DHit and Hot-reset</th>
<th>External</th>
</tr>
</thead>
<tbody>
<tr>
<td>PERST</td>
<td>External</td>
<td>Platform***</td>
<td>P-Unit, SNPS CTRL, Custom Logic</td>
<td>Asynch No Asserted only before powerup and L2</td>
<td></td>
</tr>
</tbody>
</table>

R1            | Platform   | P-Unit     | PMBUS clock | Asynch No Asserted only before powerup and L2 |          |
R2            | Platform   | P-Unit     | P-Unit*     | Asynch No Asserted only before powerup and L2 |          |
R3            | Platform   | P-Unit     | P-Unit*     | Asynch No Asserted only before powerup and L2 |          |
R4            | Platform   | P-Unit     | Custom Logic / SBR | GBR clock No Asserted only before powerup and L2 |          |
R5            | Platform   | P-Unit     | Custom Logic | GBR clock No Asserted only before powerup and L2 |          |
R6            | Platform   | P-Unit     | Custom Logic | GBR clock No Asserted only before powerup and L2 |          |
R7            | Internal   | Custom Logic | P-Unit | Asynch Yes Asserted on all SOC resets |          |
R8            | PMA, Custom Logic | Custom Logic | Asynch No Asserted only before powerup and L2 |          |
R9            | PMA, Custom Logic | Custom Logic | Asynch No Asserted only before powerup and L2 |          |
R10           | Custom Logic | Custom Logic | Asynch No Asserted only before powerup and L2 |          |
R11           | Custom Logic | Custom Logic | Asynch No Asserted only before powerup and L2 |          |
R12           | Custom Logic | Custom Logic | Asynch No Asserted only before powerup and L2 |          |
```

Mapping of Pciess resets with SOC

<table>
<thead>
<tr>
<th>Pciess Resets</th>
<th>Soc Resets</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>SoC R1_1</td>
</tr>
<tr>
<td>R2</td>
<td>SoC R2_1</td>
</tr>
<tr>
<td>R3</td>
<td>SoC R3_1</td>
</tr>
<tr>
<td>R4</td>
<td>SoC R4_1</td>
</tr>
<tr>
<td>R5</td>
<td>SoC R5_1</td>
</tr>
<tr>
<td>R6</td>
<td>SoC R6_1</td>
</tr>
<tr>
<td>R7</td>
<td>SoC R7_1</td>
</tr>
</tbody>
</table>

Reset Ordering from High Level Architecture Definition

FSDB dump of Pciess resets from PUNIT
Reset Assertion Ordering

Mapping of Pciess resets with SOC

The TX reset punit_pma_pwrgood_rst_b from the waveform is being asserted after the assertion of the all the above RX resets, thereby avoid metastability to propagate to the RX Flop.

In Reset check tool, the assertion order of these resets can be given by the following directive.

resetcheck order assert -from R1 -to R2

resetcheck order assert -from R3 -to R4
RESET DOMAIN CROSSING

- If rst1 is asserted while rst2 is not asserted, the asynchronous data from dff1 will metastability on dff2.
Evidence: Additional Matastability Risk Uncovered

6. Data crossing reset domains
Summary: Bugs found due to exhaustive reset verification in PCIE Subsystem

• First Implementation of exhaustive reset verification done on PCIE Subsystem for Client SoC
• Additional checks improved the Quality Sign Off
• Bugs Uncovered after reset verification
  – Wrong Reset Propagation
  – Power Control logic interpreted as Reset
  – Reset is always active high
  – 2nd Flop with no set/reset pin
  – Reset wrongly used as reset pin
  – Set to Set domain crossing not synchronized
  – Set to Reset domain crossing not synchronized
  – Combinational Logic in reset path
  – Async reset used in sync mode

<table>
<thead>
<tr>
<th>Partition</th>
<th>#Cell Count</th>
<th>Number of SIPS</th>
<th>Number of HIPs</th>
</tr>
</thead>
<tbody>
<tr>
<td>aux_logic</td>
<td>2577</td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td>Fabric</td>
<td>3108</td>
<td>13</td>
<td>0</td>
</tr>
<tr>
<td>PHY</td>
<td>3696</td>
<td>9</td>
<td>5(clock compensator, phy, clock control unit)</td>
</tr>
<tr>
<td>Controller</td>
<td>17554</td>
<td>16</td>
<td>18(HIPs)</td>
</tr>
</tbody>
</table>
Questions

Finalize slide set with questions slide