Scalable Reset Domain Crossing Verification Using Hierarchical Data Model

Soumya Palit, Anwesha Choudhury, Kurt Takara - Mentor Graphics, A Siemens Business

soumya_palit@mentor.com, anwesha_choudhary@mentor.com, kurt_takara@mentor.com
Introduction: What is RDC

- Data crossing from one async reset domain to another
- Transmitting (Tx) flop async-reset assertion close to clock edge can cause metastability on receiving (Rx) flop

- 'rst1' asserted very close to 'clk' posedge
- 'R2' flop output goes metastable due to setup/hold time violation
Techniques to Address RDC issues

• Reset Sequencing
  – Async-reset on Rx flop always asserts before async-reset on Tx flop
  – Rx flop already in reset state, so any change on Rx D-pin will not cause metastability

‘R2’ flop output is already 0, when Tx reset asserts
Techniques to Address RDC issues

• Isolation Techniques
  – Clockgate isolation
    • Turn off clock of Rx flop before Tx reset asserts
    • If clock is off, then any change on Rx D-pin will not cause metastability
  – Data Isolation
    • Block Tx to Rx data transmission through isolation signal before Tx reset asserts

Isolation signal is 0, turning clock of ‘R2’ to constant 0 when ‘rst1’ asserts
Need for Hierarchical RDC Verification

• SoC comprises of diverse IP blocks that are developed and verified independently

• During RDC verification of SoC, re-verification of IPs leads to redundancy and increases verification effort

• Desirable use-model is to verify integration of IPs in SoC without re-analysis of IP internals
Requirements

- Reset Logic integrity
- Accurate RDC verification
- Reset ordering
- Reset synchronizer
Requirements

- Reset logic detection across IPs
- Identify RDC issues across IP interfaces
Hierarchical RDC Analysis

• What is the Hybrid Data Model (HDM)?
  – Binary data model
  – Stores IP information
  – Guarantees RDC verification accuracy
  – Extendable to provide additional functionalities
  – Directives-based modification allowed

I am handing over IP HDM. I have also embedded integration rules in it. You will see RDC or assumption violation in case of incorrect integration.
Use Model

• Run block-level analysis
  – Specify block constraints
  – Run RDC analysis and generate HDMs
  – Review and debug block-level results

```bash
do block_ctrl.tcl
drc run block.v -d block -hrdc
```

• Run top-level analysis with block-level HDMs
  – Review and debug top-level results

```bash
resetcheck load hierdb hrdc_block.hierdb
resetcheck load hierdb hrdc_block2.hierdb
resetcheck run top.v -d top
```
Reset Logic Integrity

- Hierarchical methodology correctly detects reset logic distributed across IP
Accurate RDC verification

• RDC crossing is accurately detected across IP interface

• Provides accurate debug capabilities and shows complete path across IP interface
Reset ordering

• Seamless integration of reset ordering information
Reset Synchronizer

• Correctly detects synchronizers in reset path across IP interface
## Case Study Details

<table>
<thead>
<tr>
<th>Flat RDC Verification Methodology</th>
<th>HDM-based RDC Verification Methodology</th>
<th>Gain</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Runtime</strong></td>
<td>30min</td>
<td>10min</td>
</tr>
<tr>
<td><strong>Peak Memory</strong></td>
<td>7 GB</td>
<td>2 GB</td>
</tr>
<tr>
<td><strong>RDC Violations</strong></td>
<td>225753</td>
<td>31065</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Flat RDC Verification Methodology</th>
<th>HDM-based RDC Verification Methodology</th>
<th>Gain</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Runtime</strong></td>
<td>97min</td>
<td>60min</td>
</tr>
<tr>
<td><strong>Peak Memory</strong></td>
<td>37 GB</td>
<td>28 GB</td>
</tr>
<tr>
<td><strong>RDC Violations</strong></td>
<td>4813096</td>
<td>4247822</td>
</tr>
</tbody>
</table>
Summary

• Proposed methodology leads to accurate RDC verification with complete debug capabilities

• Ensures performance benefits

• Creates RDC IP models that can be shipped and reused across generations and SoCs
Q&A