Formal Verification &
Synopsys Verification Compiler

TVS Formal Verification Day
May 2014
Growing Demand for Smarter Verification

- Broader use of static & formal
- Much more up-front planning
- Automated setup & reuse with VIP
- Deploy critical new technologies
- Common debug across all domains
- Holistic coverage closure strategies
Current Flows Limit Smart Verification

Disjoint environments with many different tools & vendors

- Inefficient performance/productivity
- Huge costs and effort
- Different debug environments
- Deploy critical new technologies
- Different debug environments
- Who owns interoperability?
Introducing Verification Compiler

Verification Compiler

- Debug
- Static
- Formal Simulation
- Coverage VIP
Next-Generation Technologies
Static and formal verification, cross-domain debug, planning and management

- Static and formal verification
  - Property checking, LP, CDC, connectivity
- Next-generation verification IP
- X-propagation simulation at RTL
- Verification planning and management
- Advanced multi-domain debug

Next-Generation Static & Formal
Next-Generation VIP
X-Propagation Simulation
Next Generation Static & Formal Platform

- **Capacity and performance for SoCs**
  - Next-gen data model and engines

- **Compatible with implementation products and flows**
  - Usage model compatible with DC/ICC
  - Common formats, interfaces, algorithms and inference engines

- **Comprehensive set of applications**
  - Low Power, Formal, CDC, Rule-checking
  - SoC level connectivity checks, etc.

---

**3-5X capacity and performance improvements**
**excellent ease of use & debug**
Comprehensive Formal Verification GUI
Optimized for Formal Verification Debug Challenges
# Equivalence Checking Technologies

<table>
<thead>
<tr>
<th>System Level Model &gt; RTL</th>
<th>RTL to RTL</th>
<th>RTL to Gates</th>
</tr>
</thead>
<tbody>
<tr>
<td>Architectural Exploration</td>
<td>DFT/Power etc. Optimizations</td>
<td>Synthesis &amp; PnR</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Solution</th>
<th>HECTOR</th>
<th>VC Formal SEQ</th>
<th>Formality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Checking</td>
<td>Transaction Equivalence</td>
<td>Sequential Equivalence</td>
<td>Logical Equivalence</td>
</tr>
<tr>
<td>Reference</td>
<td>C (C, C++, SystemC) or RTL</td>
<td>RTL</td>
<td>RTL or Gate</td>
</tr>
<tr>
<td>Implementation</td>
<td>C (C, C++, SystemC) or RTL</td>
<td>RTL</td>
<td>Gate</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Accuracy</th>
<th>Transaction boundaries</th>
<th>Cycle-accurate</th>
<th>Boolean logic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assume Equal Inputs</td>
<td>Untimed Transaction Model A</td>
<td>Assume Equal Inputs &amp; Start State</td>
<td>Compare Boolean Fan-in Logic</td>
</tr>
<tr>
<td>Cycle Accurate Model B</td>
<td>Compare Outputs at End of Transaction(s)</td>
<td>Compare Outputs Every Cycle, Unbounded</td>
<td>Design A</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Matched Compare Points</td>
</tr>
</tbody>
</table>

HECTOR and SEQ complement Formality to enable Shift Left
Thank You