A Methodology to Port a Complex Multi-Language Design and Testbench for Simulation Acceleration

Horace Chan, PMC
Brian Vandegriend, PMC
Efrat Shneydor, Cadence
Why Simulation Acceleration

- Extra Effort
- Hard
- Easy
- RTL Simulation
- Signal Acceleration
- TBA Acceleration
- ICE
- Embedded Testbench
- Speed Up
## Benchmark Results

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Design size (in Mgates)</strong></td>
<td>26</td>
<td>38</td>
<td>50</td>
</tr>
<tr>
<td><strong>Domains used</strong></td>
<td>6</td>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td><strong>Speedup factor</strong></td>
<td>40x</td>
<td>67x</td>
<td>52x</td>
</tr>
<tr>
<td><strong>Migration Schedule</strong></td>
<td>&gt; 1 year</td>
<td>3 months</td>
<td>3 weeks</td>
</tr>
</tbody>
</table>
## 4 Types of Simulations

<table>
<thead>
<tr>
<th>Type</th>
<th>Where the RTL runs</th>
<th>Objectives</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal</td>
<td>RTL Simulator</td>
<td>TB profiling, optimization Generate golden reference</td>
</tr>
<tr>
<td>SA_SIM</td>
<td>RTL Simulator</td>
<td>Debug SCEMI-pipe, Debug DPI integration Clean up TB-RTL binding</td>
</tr>
<tr>
<td>SA_SW</td>
<td>SW model of the Box</td>
<td>Debug RTL synthesis flow Setup shell scripts for SA</td>
</tr>
<tr>
<td>SA_HW</td>
<td>The Box</td>
<td>Debug mismatch between SW model and the Box Benchmark SA</td>
</tr>
</tbody>
</table>
Compile the RTL

If the RTL won’t compile, there is zero speedup

• Every VHDL/Verilog compiler is not the same
• Use the same tool chain from the same vendor
• Don’t hack the RTL to work around tool problems
  – fix the tool!
• Use the same workspace as normal simulation
• Reuse compile scripts from ICE flow
• Use SV config to swap in TBA components
Testbench Enhancement

• Testbench is the bottleneck
  – Transaction based processing is a must
  – TB should use less than 2-3% CPU cycle

• Clean interface between TB and RTL
  – No wait delay
  – No direct RTL signal access
  – No backdoor access
  – No cycle by cycle interaction
Testbench Enhancement

Specman makes the TB upgrade easy

- Use AOP to keep TB code for normal simulation and acceleration in two different aspects
  - No messy factories or callback functions

- Use powerful macros and reflection interface to catch unknown TB-RTL interaction
  - Build debug tools to inspect the TB code
Regression Management

- Use the same regression tool as normal simulation
  - SA is like a super fast computer
- Download snapshot to the box is slow
  - Download once, run all the sims in the same session
- Auto re-launch overnight failed sims
  - SA_HW is too expensive for interactive debug
  - Hot swap to SA_SW before the failure point
- No checkpoint save and restore in SA (yet)
Conclusion

• Focus on quick bring up of SA

• Don’t optimize for the best possible speed up
  – SA can never run as fast as ICE

• Positive ROI around 30-40x speed up*
  – In terms of raw simulation through put
* depends on the license fees discount and acceleration box lease contract

• Faster debug turnaround time = higher productivity