Co-simulation platform of SystemC and System-Verilog for algorithm verification

Li Jinghui, Xi’ An R&D Center, Inspur, Xi’an, China (lijinhui01@inspur.com)
Shao Haibo, Xi’ An R&D Center, Inspur, Xi’an, China (shaohaibo@inspur.com)
Gou Jiazhen, Xi’ An R&D Center, Inspur, Xi’an, China (goujiazhen@inspur.com)
Agenda

• Current Verification Challenges
• Why SystemC
• The SystemC model of compression algorithm
• Verification with SystemC
• C/C++ vs. SystemC model for Verification
• Advantages of Co-sim Platform of SystemC and System-Verilog
Current Verification Challenges

- increasing complexity of SoC design
- time to market is getting shorter and shorter
- verification growth in cost and schedules

Source: IBS Global Semiconductor Industry Service Report, 2018
Why SystemC

• extended library of C++
• hardware-oriented features
  – Time model
  – Hardware data types
  – Module hierarchy to manage structure and connectivity
  – Concurrency model
  – Communications management between concurrent units of execution
Why SystemC

• Support modeling at different levels of abstraction
  – ALM
  – SAM
  – TLM
  – RTL
• highly reused in the development of SoC
• as a bridge between various departments
Why SystemC

- C/C++ vs. SystemC for development processes
The SystemC model for compression algorithm

• algorithm requirements
  – Original algorithm is to find a possible matching character byte by byte, can only process one byte of input data per cycle
  – Need to increase the data throughput rate, explore the parallelization of hardware implementation, process more data in one cycle
  – Limited memory resources, HW architecture has no input buffer
  – Pipelined processing methods are used to implement a forward and branch-free algorithm
The SystemC model for compression algorithm
The SystemC model for compression algorithm performance analysis

<table>
<thead>
<tr>
<th>File</th>
<th>QAT</th>
<th>initial arch</th>
<th>updated arch</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>SRAM 240KB</td>
<td>SRAM 1152KB</td>
<td>SRAM 576KB</td>
<td>SRAM 288KB</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ratio BW12_C12</td>
<td>ratio BW11_C12</td>
<td>ratio BW10_C12</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>dickens</td>
<td>1.659</td>
<td>1.449</td>
<td>1.58</td>
<td>1.652</td>
<td>1.592</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mozilla</td>
<td>2.188</td>
<td>1.853</td>
<td>1.966</td>
<td>1.953</td>
<td>1.922</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mr</td>
<td>1.992</td>
<td>1.754</td>
<td>1.91</td>
<td>1.905</td>
<td>1.875</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>nci</td>
<td>5.765</td>
<td>2.681</td>
<td>4.36</td>
<td>4.356</td>
<td>4.34</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ooffice</td>
<td>1.618</td>
<td>1.453</td>
<td>1.53</td>
<td>1.521</td>
<td>1.489</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>osdb</td>
<td>1.89</td>
<td>1.710</td>
<td>2.22</td>
<td>2.131</td>
<td>1.86</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>reymont</td>
<td>2.207</td>
<td>1.736</td>
<td>1.96</td>
<td>1.941</td>
<td>1.902</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>samba</td>
<td>2.889</td>
<td>2.079</td>
<td>2.73</td>
<td>2.696</td>
<td>2.61</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sao</td>
<td>1.158</td>
<td>1.188</td>
<td>1.18</td>
<td>1.173</td>
<td>1.162</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>webster</td>
<td>2.165</td>
<td>1.808</td>
<td>2.18</td>
<td>2.154</td>
<td>2.092</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x-ray</td>
<td>1.136</td>
<td>1.057</td>
<td>1.124</td>
<td>1.107</td>
<td>1.072</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>xml</td>
<td>3.714</td>
<td>2.641</td>
<td>3.73</td>
<td>3.875</td>
<td>3.772</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>average</td>
<td>2.247249</td>
<td>1.823031</td>
<td>2.131843</td>
<td>2.119311</td>
<td>2.061693</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Verification with SystemC
Verification with SystemC

• Build a pin-level algorithm model

- The interface timing of the model is exactly the same as the RTL;
- Output intermediate results through the SystemC out port for debugging
Verification with SystemC

• Use the SystemC model to build UVM TB in advance
  – Before RTL is available, the SystemC model can be used as a DUT to build TB for preliminary verification
Verification with SystemC

- Verify the real RTL
- Upgrade the verification environment
Verification with SystemC

- EDA tools such as VCS has built-in support for SystemC.
- When integrating the SystemC model into the System-Verilog environment, just instantiate the model like an RTL module.
Verification with SystemC

- Co-sim platform of SystemC and System-Verilog

```
compress_top_tb
```

```
reference model
```

```
systemc model
```

```
COMPRESS_UVM_ENV
```

```
in_agent
  seqr
  driver
```

```
out_agent
  monitor1
  monitor2
  ...
```

```
comp_in_if
comp_hash_if
comp_merge_if
comp_out_if
```

```
seq
```

```
DUT
```

```
RTL
```

```
scoreboard
```

```
... SC RTL ...
```
Verification with SystemC

- simulation command with VCS

```
//Compile SystemC:
syscan –cpp g++ -cc gcc –cflags –g –full64 $(SC_SRC)

//Compile System-Verilog:
vlogan –full64 –sverilog $(SV_SRC)

//elaboration
vcs –full64 –cpp g++ -cc gcc -sverilog –sysc SV_TOP_TB
```
Verification with SystemC

- simulation results
Verification with SystemC

- debug SystemC model with VCS Cbug
Verification with SystemC

- VCS SystemC co-simulation interface
  - Enables Verilog, VHDL and SystemC modeling to work together
  - Supports for Verilog-top/SystemC-top/multiple-top topology
  - VCS Extensions to SystemC Library, such as get_full_name()
  - Unified compilation/simulation/debug flow
  - Transaction Level Interface (TLI) enables integrating transaction level SystemC models into a SystemVerilog environment seamlessly and efficiently
Advantages of Co-sim Platform
C/C++ vs. SystemC model for Verification
## C/C++ vs. SystemC for Verification

<table>
<thead>
<tr>
<th>items</th>
<th>C++</th>
<th>SystemC</th>
</tr>
</thead>
<tbody>
<tr>
<td>integration</td>
<td>Develop DPI-C function</td>
<td>As simple as instantiating RTL</td>
</tr>
<tr>
<td>driver</td>
<td>Call the DPI function in the test case</td>
<td>Driven by the same driver of RTL</td>
</tr>
<tr>
<td>checker</td>
<td>Need to store the results of C/C++ model&lt;br&gt;Take up additional storage space</td>
<td>Monitor the results through interface</td>
</tr>
<tr>
<td>debugging</td>
<td>Internal variables are difficult to debug&lt;br&gt;Dump variables to files or use breakpoints</td>
<td>Output key variables through SystemC ports or signals</td>
</tr>
</tbody>
</table>
Verification start earlier

Verification can start at least 1 month in advance
The reusability of SystemC model is high.

Be reused by Arch/HW/SW team

Be easily reused for module/sub-system/SOC verification

When the RTL of some modules has not been delivered, SystemC model can be used to start sub-system or SOC level verification.
## Summary

<table>
<thead>
<tr>
<th>items</th>
<th>C/C++ Model</th>
<th>SystemC Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model integration time</td>
<td>About 3 days</td>
<td>Less than 1 day</td>
</tr>
<tr>
<td>Debug</td>
<td>difficult</td>
<td>easier</td>
</tr>
<tr>
<td>Verification start earlier</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Verification reusability</td>
<td>low</td>
<td>high</td>
</tr>
</tbody>
</table>
THANK YOU!