Using Simulation Acceleration to achieve 100X performance improvement with UVM based testbenches

-Ranjith Kumar K
Agenda

• Why Simulation Acceleration
• Signal based vs Transaction based
• TBA Env Setup
• Migration to TBA
• Advantages of TBA
• Improving Simulation throughput
• Debugging methods
• Performance improvement results
Why Simulation Acceleration

• Takes long hours to simulate few hundreds of packets at SoC level.
• SA (TBA) will complement simulation, focus testing on areas where simulation time is prohibitive. Long tests will be the primary focus area in Acceleration
• Acceleration can achieve >100X performance improvements
• Reuse part of the TB between Simulation and Acceleration
Signal Based Acceleration

• Split the UVM Testbench and DUT
• Testbench runs on the SW-simulator
• Synthesizable DUT runs on the HW-accelerator

• Issue:
  • Performance improvement is limited by excessive context switching due to signal level communication between UVM testbench and DUT

• Can be 5-50X Faster
Transaction Based Acceleration (TBA)

- TBA moves part of the testbench into the hw-accelerator
- Infrequent and Information-rich data.
- Interface between the simulator and the hw-accelerator consists of function calls rather than individual signal transitions
- Reduces the context switching or frequency of synchronization between simulator and the hw-accelerator
- May reduce the amount of data to be transferred
- Can be 50-500X Faster
Simulations Timing Profile

- Pure SW simulation
- Complete TB in SW, DUT in HW (Signal Based Acceleration)
- TBA (Transaction Based Acceleration)
Typical UVM TB Env Setup

- **TB Top**
  - Clock/Reset
  - Driver _0
  - Driver _1
  - driver_n
  - SR
  - Scoreboard
  - Memory

- **Simulation Model**

- **DUT**
  - Signal Synchronization time (occurs at every clock/event)
TBA Env Setup

Simulation Model

SW

- SW Proxy
- SW Driver_0 Proxy
- SW Driver_1 Proxy
- ISR
- Scoreboard

HW

- HW Proxy
- Clock/Reset
- HW Driver BFM
- HW Driver BFM
- HW Driver BFM

DUT_TOP

Memory
TBA : SW – HW Communication

- Virtual Interface
- SystemVerilog DPI
- SCE-MI (Standard Co-Emulation Modeling Interface) : efficient for streaming
Migrating to TBA

- Testbench partition : 2 TOP
  - SW_TOP : runs on simulator, untimed, HW-BFM proxies, stimulus generation, scoreboard, no #delays, no XMR, no @wait (signal) etc
  - HW_TOP : runs on accelerator, timed, all BFMs accessing DUT signals, clk & reset generators etc
  - Transaction based communication between SW_TOP & HW_TOP
Migrating to TBA contd...

• Simulation VIP ➔ Acceleration VIP
  • Industry standard AVIPs, Mem models (DDR/Flash etc) available from vendors
  • Synthesizable BFMs for the in-house VIPs e.g. SHELL
  • No analog block, PLL, serdes or PHY etc in the DUT/TB.
  • Backdoor access mechanisms (Peek/Poke) for memories
Context Switching

2 context switches
1 marg call
per CMD processing
Bundling of Commands/Completions

2 context switches
1 marg call
per 100 CMDs
Performance tuning

• 3 components that consume wall clock time
  • Time used in SW testbench execution => Minimize
  • Time used in context switching => Minimize
  • Time used in HW execution => Maximize

• Minimize SW testbench execution time
  • Thin testbench
  • Reduce the memory footprint in the testbench, HeavyPkt- >LitePkt
  • Avoid unnecessary randomization/function calls etc
  • Profiling / TB architecture review etc
Performance Tuning Aspects

• Minimize the amount of data transfer between HW-SW
  • Infrequent and information-rich data
  • Start_Pattern / Incr_Pattern
  • Instead of sending the wr_data along with the cmd from sw->hw, we can send random start_pattern / incr_pattern with each cmd
  • The data can be generated in the hw itself before driving it to the DUT
  • Minimize the backdoor accesses (e.g. marg_put/marg_get) calls
• Scoreboarding in HW, as applicable
• Tool provides various techniques to improve the performance:
  • Upgrade from 2x to 1x mode (both edges)
  • Increase FCLK frequency=> analyze design for long combinatorial paths etc
Debugging

- Lot of time is spent on debugging
  - TB issues
  - Dut issues
  - Env issues
- Efficient debugging technique is very important to root cause the issues.
- Debugging tools
  - Waveforms – Takes lot of time to dump from emulator’s buffer to host side.
  - Log files – Verbosity levels.
- It may require multiple runs to root cause and fix the issues.
- Efficient techniques
  - Recording SW and/or HW side activities and replaying.
  - Dumping selective signals.
  - Limited duration dumping.
HW debug

• Capture the stimulus at every interval or relevant interval.

• Replay the sim using the captured stimulus and dump the waveform in the required duration.
SW debug

- Capture the HW activity till point of interest. Save the HW state.
- Re-run without attaching the Emulator.
- Attach, restore the HW and run as usual.
## Performance improvement results

<table>
<thead>
<tr>
<th>Data Transfer Size</th>
<th>Simulation wall clock time</th>
<th>Emulator wall clock time</th>
<th>Speed Up (X factor)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1KB-64KB<em>100</em>8</td>
<td>25888sec</td>
<td>118 sec</td>
<td>219</td>
</tr>
<tr>
<td>64KB-256KB<em>100</em>8</td>
<td>46694sec</td>
<td>174 sec</td>
<td>268</td>
</tr>
<tr>
<td>512KB-1MB <em>100</em>8</td>
<td>43637sec</td>
<td>111sec</td>
<td>393</td>
</tr>
</tbody>
</table>

© Accellera Systems Initiative
Summary and Advantages of TBA

• Most active part of the testbench (BFM/monitors) runs in the hardware like an actual ASIC.
• Offers the re-usability of the verification testbenches
• Allows leveraging of the advanced features in the HVL, CDV methodologies e.g. UVM.
• Best of both the worlds :
  • HW => the desired speed,
  • SW => the required flexibility
• Assertions are supported
• Simulator-like debug/wave-viewing environment
• Helpful in faster Coverage/Verification closure and hence enhancing overall productivity
Questions?