The Art of Portable and Reusable UVM Shared System Memory Model
Verification Methodology across Multiple Verification Platforms
Roman Wang, Thomas Bodmer, Beryl Chen
Advanced Micro Devices, Inc.

Introduction
The shared system memory is an essential design element in both IP and SoC. A comprehensive system memory model (SMM) is critical to simplify the design verification process while ensuring the quality. In different verification platforms, SMM is used to create test sequences, predictor, scoreboard, etc. With more and more companies adopting the UVM methodology to develop their verification environment from IP level to SoC, maximal reusability is always the goal. When trying to make vertical or horizontal reuse of verification environment across IP, SoC, teams and projects, SMM reusability can be a great challenge. In this paper, this portable and reusable UVM SMM is presented. It can be reused without any changes across various verification platforms, such as IP UVM standalone, UVM IP, Mega IP Combo Whacker (UVM CW, a kind of AMD specific subsystem verification in SoC database), virtual FPGA (vFPGA), a kind of simulation environment using synthesizable design and glue logics for FPGA), and SoC full chip.

Challenges
• Different verification teams or groups deploy and adopt the different SMM written by different languages, such as Verilog/VHDL, System Verilog, E, Vera, or C++/C.
• It’s really difficult to make different SMMs work together in a specific verification platform.
• Only few SMM implements the memory allocation features. Ex. It can (de-)allocate or resolve exclusive memory regions.
• Potential PLI/DPI usage will affect verification performance a lot.
• Different 3rd part UVM VIP provides the slave responder which makes the memory verification hard.
• Potential PLI/DPI usage will affect verification performance a lot.
• Different 3rd part UVM VIP provides the slave responder which makes the memory verification hard.

The methodology is made of five key concepts:
1. UVM shared system memory export (SSMM), it is a System Verilog class and supports a singleton instance. It's designed as an export proxy to provide multiple friendly easy to use APIs for end-users to communicate GMEM.
2. UVM generic save restore memory model (GMEM), it is a System Verilog class and supports a singleton instance. It is designed to save and restore memory data with the address as index.
3. UVM unique slave memory sequence (USMSEQ), it is a unique parameterized UVM sequence which is protocol independent and co-work with SSMM.
4. UVM memory knobs container (MKC), it is a UVM object and has types of local associate arrays to support knob layer control.
5. UVM memory export adapter (MEA). It is a UVM object and derives from uvm_reg_adapter to implement the protocol specific data translation.

Figure 3 presents how the SSMM, GMEM, USMSEQ, MKC, and MEA are used in a typical UVM environment.

Working Models
• Traditional built-in memory model in UVM VIP (Figure 4).
• How USMSEQ works with other Slave UVM VIP? (Figure 5).
• How the protocol memory export adapter working with memory slave sequence? (Figure 6).
• Front-door use model in IP level (Figure 7).
• Front-door use model in SoC level (Figure 8).

Prerequisites
• System Memory Address Alignment
• Front-door use model in IP level (Figure 7).
• Front-door use model in SoC level (Figure 8).

Conclusion
Based on our numerous successful UVM projects’ experience for years, the proposed portable and reusable UVM shared system memory model verification methodology and its typical use model has been very effective in verifying different level of system memory scenarios. The memory export provides many easy-to-use APIs to ease the implementation. The export adapter library provides a quick connection to the different bus protocols. The protocol independent unique memory save sequence bridges the gap in adoption of different protocol VIPs. It can be portable and reusable across multiple verification platforms.

Contact information
Roman Wang
Design Verification Architect
Roman.wang@amd.com  @ AMD Shanghai, China
Thomas Bodmer
Sr. Manager
Thomas.Bodmer@amd.com  @ AMD Sunnyvale, U.S.
Beryl Chen
Design Verification Architect
Roman.wang@amd.com  @ AMD Sunnyvale, U.S.

Table 1 presents the task and efforts in IP or Mega IP level verification.
Table 2 presents the task and efforts in UVM Combo Whacker or SoC level verification. It’s clear to see the effort is dramatically reduced for 2nd IP.

Table 1
<table>
<thead>
<tr>
<th>Task</th>
<th>1st IP Efforts</th>
<th>2nd IP Efforts</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. IP Methodology Integration</td>
<td>1 day</td>
<td>0.5 day</td>
</tr>
<tr>
<td>Smoke test with debug</td>
<td>12 hours</td>
<td>4 hours</td>
</tr>
</tbody>
</table>

Table 2
<table>
<thead>
<tr>
<th>Task</th>
<th>1st IP Efforts</th>
<th>2nd IP Efforts</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. IP Methodology Integration</td>
<td>1 day</td>
<td>4 day</td>
</tr>
<tr>
<td>Smoke test with debug</td>
<td>12 hours</td>
<td>12 hours</td>
</tr>
</tbody>
</table>