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 ensure 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 great challenges. 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 or 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 synthesible 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 has different built-in SMM implementation.

•The legacy SMM can not easily work with UVM memory front door operation APIs.

•The flexible debug abilities and comprehensive errors injection are all required.

• SMM reuse is a big problem from IP to SOC .

The generic UVM SMM developed by AMD South Bridge (FCH) group addresses all the above concerns.

Prerequisites

Alignment type	Description
0	Half-Word alignment (16bits)
1	Word alignment (32bits)
Up to 12	1024 Words alignment (4KB)

Figure 1

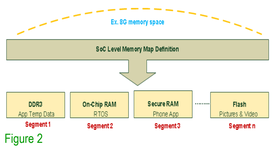

Figure 2

- *System Memory Address Alignment*

Figure 1 presents the supported alignment for AMBA bus system. Ex. The Word aligned address needs the address[1:0] to be zero.

- *System Memory Segments*

In a complex SoC design, it usually defines several different types of memory which attached on the system bus. They are shared with most of IPs and playing the different roles in different application. Figure 2 presents a clear picture of the relationship between system memory and memory segments.

Big picture of the concepts and key elements

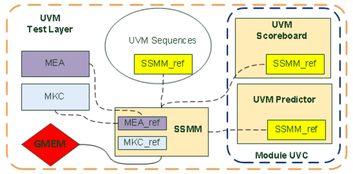

Figure 3

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 an UVM object and has types of local associate arrays to support knob layer control.
5. UVM memory export adapter (**MEA**). It is an 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).

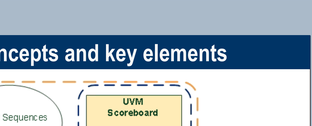

Figure 4

❖ *How USMSEQ works with other Slave UVM VIP?* (Figure 5).

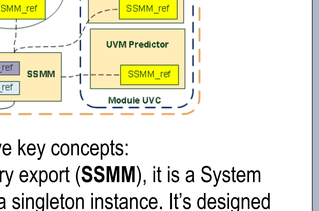

Figure 5

❖ *How the protocol memory export adapter working with memory slave sequence?* (Figure 6).

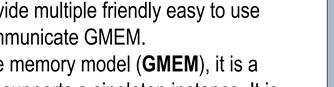

Figure 6

❖ *Front-door use model in IP level* (Figure 7).

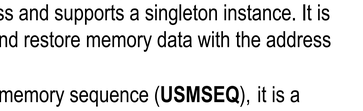

Figure 7

❖ *Front-door use model in SoC level* (Figure 8).

Figure 8

IP UVM Stand-alone Verification Platform Diagram

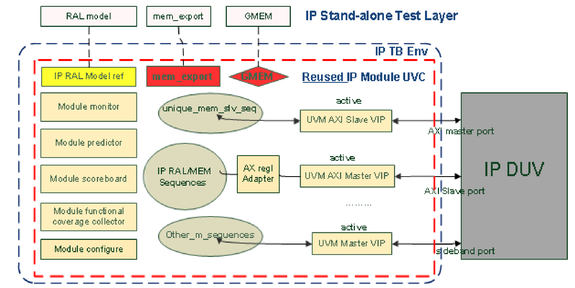

Figure 9

IP Virtual FPGA Verification Platform Diagram

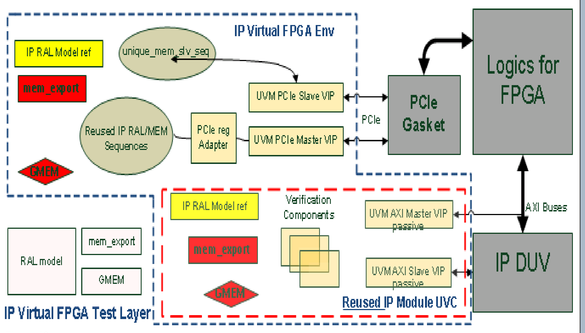

Figure 10

UVM IP Combo Whacker SoC Verification Platform Diagram

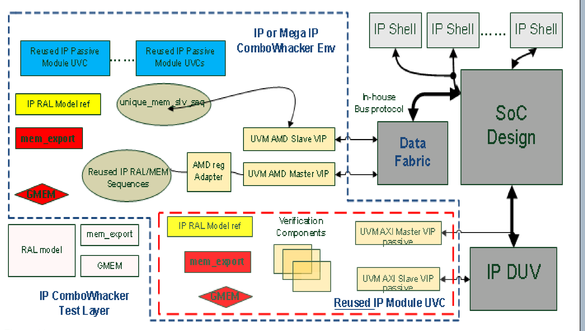

Figure 11

SoC Full Chip UVM Verification Platform Diagram

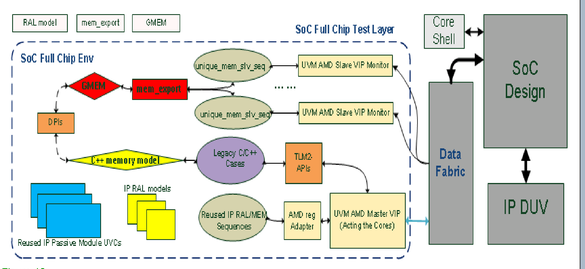

Figure 12

Research Poster Design Services

Our portable and reusable UVM shared system memory model verification methodology has been across numerous UVM projects in our group for more than three years. Typically we can see such projects obtaining an approximate 96% effort reduction compared with creating the memory model methodology from scratch and reusing them during adopting the UVM VIPs with different protocols across different verification platforms.
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.

IP/Mega IP level	Task	Efforts per person
1st IP	Methodology Integration	0.5 day
	Smoke test with debug	1 day
2nd IP	Methodology Integration	0.4~0.6 hour
	Smoke test with debug	1 hour

Table 1

UVM CW/SoC level	Task	Efforts per person
1st IP	Methodology Integration	1 day
	Smoke test with debug	4 day
2nd IP	Methodology Integration	1 hour
	Smoke test with debug	12 hours

Table 2

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 salve 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.