Distributed Simulation of UVM Testbench

Theta Yang (theta.yang@amd.com)
Advanced Micro Devices
Bldg33 River Front Harbor, Zhangjiang Hi-Tech Park, Shanghai, China

Problem Statement

Extra workload is involved in both IP and SOC testbench development to support IP to SOC testbench reuse. And simulation performance factors prevent the full leverage of the verification component from IP level to SOC level.

- IP's verification component needs to support configuration options what are not really needed at the IP level. For example, an alternative passive mode.
- Instantiation, configuration and connection of IP-level UVM component at the SOC level needs significant work effort and debug time.
- IP UVM components are compiled and simulated, this results in huge memory usage for the SOC simulation and considerable degradation in the simulation speed.

By running multiple simulation nodes in parallel and exchanging internode TLM messages through inter process communication (IPC), the SOC level simulation computing load is distributed and simulation performance gets boosted.

Benefit of the Distributed Simulation System

1) Simulation performance is scalable

SOC-level testbench is partitioned and distributed into multiple IP level testbenches, where the computing load and memory size required by SOC level testbench is reduced. Each IP-level simulation requires less computing performance and memory size. The framework leverages the multi-thread capability of a modern simulation server, where each IP has a separate simulation instance and the number of simulation processes can be allocated based on the required simulation performance.

2) Fully reuse IP-level testbench at SOC level

SOC-level simulation testbench is composed of a simulation cluster of IP-level simulation testbenches. The IPC adapters are extra verification components that can be added to UVM-based testbench without changing the existing testbench. The original verification components such as stimulus driver's and checkers are still functional as usual. The distributed simulation framework can also support an IP testbench implemented in different simulation language, for example a

UVM-based simulation can be connected with a SystemC based simulation.

3) IPC adapter design is generic and can be used by various interfaces and in multiple projects

The IPC adapters for standard interfaces such as SDP (AMD proprietary system bus) and AMBA AXI can be reused for multiple projects. The communication carrier can be either inter process communication or network socket provided by computer operation system, the internal FIFO size is parameterized, and the payload type of the communication follow generic payload defined in standard SystemC TLM2. Since it is based on well defined industrial standard it is easy to develop an IPC adapter for other interface protocols.

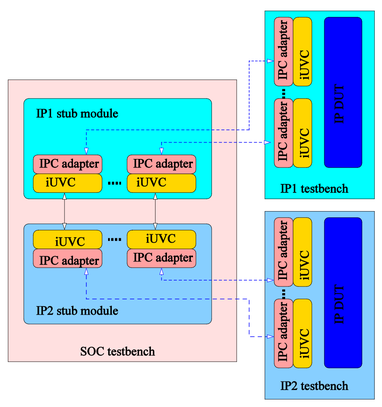
Testbench Partition and Topology Mapping

SOC-level simulation is composed of a cluster of simulation instances among which one of the instance simulates the SOC DUT and its testbench. Each of the other instances simulates an IP DUT together with its testbench.

IPC adapters are connected in point-to-point style in order to relay the TLM transaction from interface UVC in SOC simulation node to interface UVC residing at the distributed simulation node.

All of the IPC connections are between SOC testbench and IP testbenches, and there is no direct connection between the various IPs' simulation nodes. The IP testbenches are wrapped by a stub module at SOC testbench, the stub module and the distributed IP testbench simulation instances add together to perform the IP's functionality in SOC-level simulation.

The architecture of the proposed distributed testbench is demonstrated in Fig.1.



(Figure 1)

IPC Adapter Design

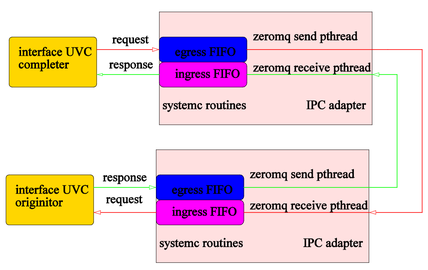
When interface UVC is a transaction completer, it accepts a transaction request from DUT's originator interface and sends the request to IPC adapter. The IPC adapter is responsible to buffer the request and send it to external components through IPC networking endpoint.

When the DUT is a transaction completer, the IPC adapter receives the transaction request by external IP through the IPC endpoint and forwards the request to interface UVC which will perform as a traffic originator.

The IPC adapter provides DPI routine to be used by System Verilog based interface UVC to send/receive transactions.

Each IPC adapter forks two background processes to send and receive transactions through ZeroMQ API. The receiving thread is a "server" which accepts the connection and request from a "client' of the remote connected IPC adapter.

The IPC adapter is designed as a generic building block, different transaction type and FIFO depth are all parameters that can be specified when it is instantiated.



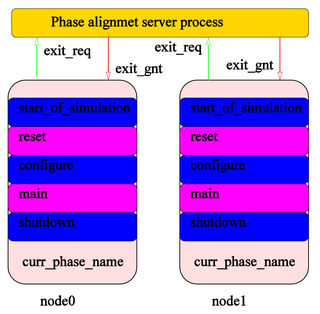
(Figure 2)

Simulation Phase Alignment

Each IP testbench needs to be setup properly and initialized before it can originate or respond to the transaction at the IPC adapter. And all IP testbenches are required to reach the functional mode before the cross node TLM transaction can be exchanged.

A central server process is used to sync the simulation phases between simulation nodes. All sub phases under UVM simulation run phases are synchronized between simulation nodes as shown in Fig. 3.

When a simulation process wants to exit a specific simulation phase, it will send phase sync message to the phase server. The phase server will acknowledge the simulation phase requestor after the request for all simulation nodes are received.



(Figure 3)

The phase alignment algorithm can be further extended to support a global slow clock, we can make all simulation instances run synchronously on the slow clock. With this global clock, when a simulation instance tries to run the simulation ahead, it will get suspended and waits for the slower simulation node.

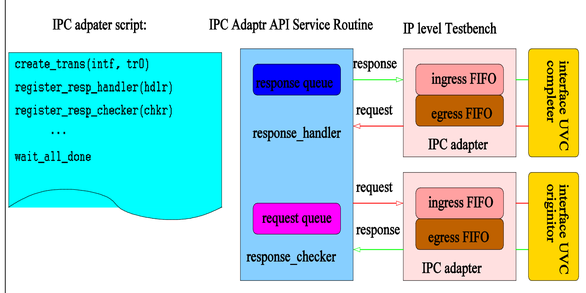
With the global clock, all the cross node traffic can be constrained to complete in the same cycle. With this extension, the simulation time deviation between simulation nodes can be limited to the single cycle of the slow global clock.

Bring the script capability to UVM testbench

The distributed simulation frame work can also be used in a single node simulation environment. Besides to connecting the two simulation nodes, we use a simulation control script to control the IP simulation to develop debug.

A backend service routine in C++ talks with the IPC adapter in the simulation node through the ZeroMQ API. The routine contains a queue to buffer the transaction request or response to be sent to the ingress FIFO of an IPC adapter. And it also contains the list to record the callback routine to be called when a response or a request is received from an IPC adapter.

The script currently support generating transaction, registering callback for response handler or response checker function.

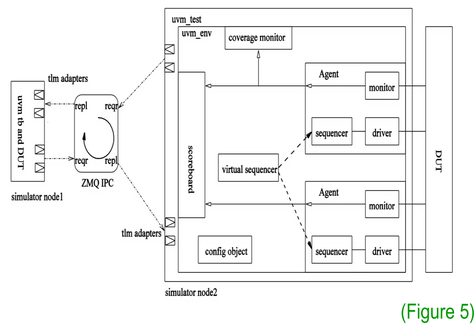


(Figure 4)

Prototyping System Simulation Performance

The top level of the prototyping system for the distributed simulation framework contains two simulation nodes, one simulates SOC level testbench and another simulates IP level testbenches.

Both the simulation nodes had testbench already coded in the UVM implementation. A special UVM test is developed for each UVM testbench, and they are running separately on each simulation node when the distributed simulation is launched.



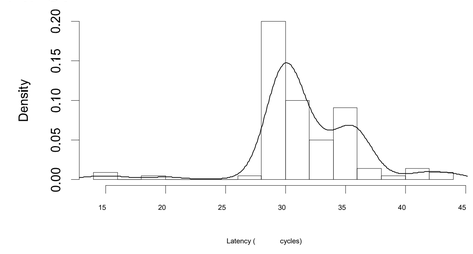
(Figure 5)

The below Table I shows the results including compile time and run time performance factors. Both the memory used to compile the simulation image and the memory used to run the simulation is significantly reduced. The average simulation speed is increased from 5.4 ns sim speed per seconds to 9.6 ns sim speed per second and it makes the simulation much faster.

TABLE I PROTOTYPING SYSTEM PERFORMANCE FACTORS

Configuration	Compile Peak Mem(M)	Compile Avg. Mem(M)	Compile Time(S)
Single image sim	13378	8385	5548
DIST sim SOC node	8968	5674	4111.6
DIST sim IP node	4978	4820	5654.9
	Sim Peak Mem(M)	Sim Avg. Mem(M)	Sim Speed(ns/s)
Single image sim	8884	7966	5.4
DIST sim SOC node	5458	4799	9.6
DIST sim IP node	4978	4820	8.1

The transaction transport time is the absolute latency involved in the distributed simulation architecture to send transaction from one simulation node to another. It is asynchronous to the simulation time modeled in each simulation node. From a specific simulation node, the latency is presented as the cycle number needed for sending the transaction request at the egress FIFO to the receiving time of the transaction response at ingress FIFO.



(Figure 6)

Limitation of Proposed Distribute Scheme

The proposed simulation framework doesn't have cycle accurate simulation timing at the IP boundaries. The transaction buffer and inter-node communication causes extra transaction latency between IP's. It cannot replace the traditional SOC level simulation which requires simulation to be performed at signal level or bus protocol level.

The simulation framework only applies to SOCs which use latency-tolerate interface for IP connectivity. And as it involves extra latency at IP boundaries, the simulation framework is not suitable to run performance measurement tests.