Utilization of Emulation for accelerating the Functional Verification Closure

Varun Kumar C, Senior Engineer, SSIR, Bangalore (varun.k2@samsung.com)
Sekhar Dangudubiyyam, Associate Director, SSIR, Bangalore (sekhar.d@samsung.com)
Madhukar Ramegowda, Associate Technical Director, SSIR, Bangalore (madhukar.r@samsung.com)

Introduction

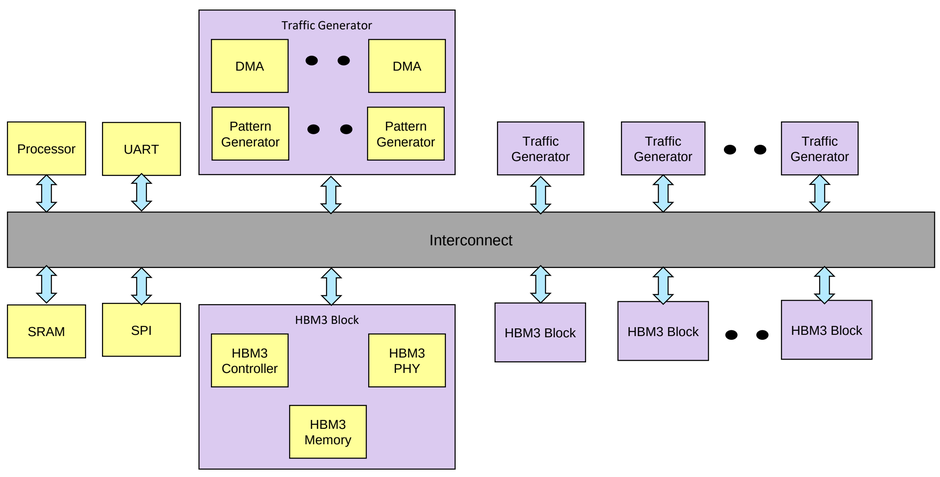
With the increasing complexity of SOC designs for AI, automotive, wearables and mobile applications, the focus on creating robust system stimulus covering all critical interfaces and measurement of functional coverage at SOC level has increased significantly. IP/Sub-system level DV scope covers internal design core with synthetic traffic using bus transactor and often do not see simulation runtime challenges, when the same IP/Sub-system is integrated in SOC, generating real traffic from higher level abstraction poses limitation in covering varieties of scenarios predominantly due to longer runtimes (i.e. verification at SOC level requires compulsive clock initializations and memory initializations). Verification of the SOCs with memory models like HBM3 and multiple instances of Traffic Generators consists of complex concurrent traffic scenarios. Implementing such large data traffic scenarios on the emulation systems which provides rich debug environment and support for functional coverage analysis becomes absolutely necessary, since simulation poses challenges like very long runtimes, complex testbench setup, various initialization requirements, unsteady license availability and license holding, hence blocking us from performing multiple iterations with various functional and address coverage.

Proposed Methodology

Based on the requirement for implementing the concurrent traffic scenario on the emulation system, below objectives were defined :

- To use the emulation methodology for accelerating verification of multimedia, high speed IO and performance verification.
- Implementing complete functional coverage solution with powerful single binding statement providing coverage metrics for all the instances of DMAs and Custom DMAs.
- Complete leverage of existing UVM SV environment for preparing software workloads.
- Use this emulation flow for self checking and UART aided debug.
- Developing software like interrupt service routines for SOC interrupts handling.
- Periodic interrupts from the Multi Core Timer(MCT) for printing the runtime units.
- T32 interface for live monitoring of various IPs and memory models in SOC.
- Developing an emulation image with the SPI transactor for alternate bootflow.
- Capture the waveform for the desired time window of the concurrent activity.

Implementation Details

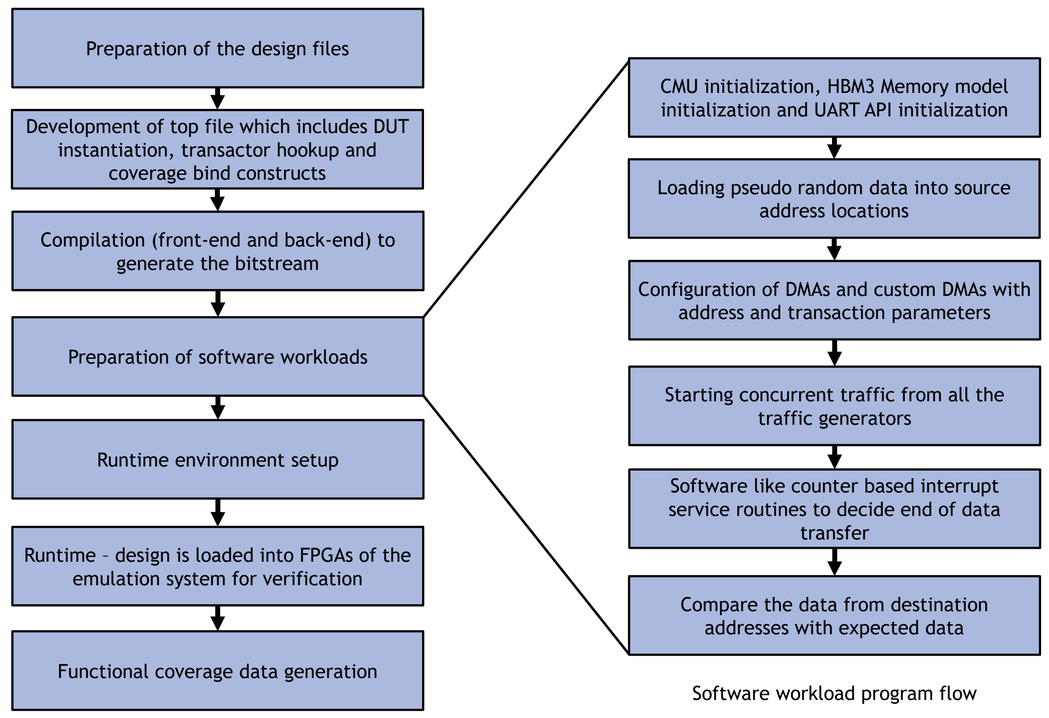


Architectural overview of HBM3 Controller

Above diagram represents the architectural overview of HBM3 Controller.

SRAM memory is used for loading software workloads, which is accessed by the processor. UART API's in the software make use of UART IP and its transactor for message display. SPI IP provides SPI interface for alternate bootflow. We have multiple instances of traffic generators, in each traffic generator we again have multiple instances of DMA330 and the custom DMAs all functioning to generate the traffic for the HBM3 block.

Implementation Flow Chart



Verification on Emulation systems

Software workload program flow

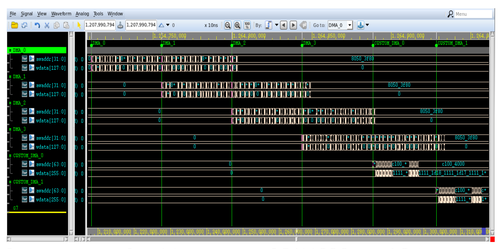
Results Table

Runtimes were captured for concurrent traffic scenarios, both in simulation and emulation environments and the data is presented in the below table,

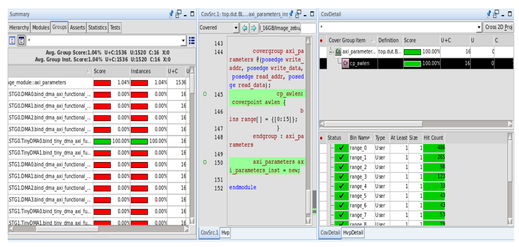
Events	RTL Simulation	Gate Level Simulation	RTL Emulation	Runtime Efficiency Improvement
HBM initialization	69 hours 21 mins	415 hours	<30 secs	8000x ~ 50,000x
DMA Verification	15 hours 32 mins	163 hours 13 mins	<10 mins	100x ~ 1000x
Custom DMA runs	13 hours 50 mins	133 hours 49 mins	<7 mins	100x ~ 1000x
1 ms timer run	10 mins	50 mins	<10 secs	60x ~ 300x

Runtime comparisons between simulation and emulation

Waveform captured and Functional coverage data collected are given in the figure below,



Waveform capture for 4 instances of DMA330s and 2 instances of Custom DMA



Functional Coverage implementation on Emulation Platform

Conclusion

The methodology discussed has been successfully validated on emulation platform with waveform capture and complete debug capabilities. The future work that has been identified for this methodology are :

1. Extending to other High Speed IPs like PCIE, USB etc. Use existing protocol transactors and cover larger traffic test like DMA scenario.
2. Backdoor read/write and memory dump operations of emulation system can be used for data capture and comparison.
3. Testbench infrastructure can be scaled up to netlist based runs for zero delay GLS. PnR ready netlist based emulation is already proved methodology.

Hence, scaling up the usage of emulation infrastructure in accelerating functional coverage closure of critical interfaces in SOC like power management, DRAM, fabric acceptance/issuing capabilities, coherency, memory management (MMU) and Q/P channel handshake et al. This brings together various aspects of simulation and emulation like hardware implementation of SOC design, peripheral transactors, software workloads, functional verification, functional coverage model and covergroups, trace32 based debug and waveform capture, ultimately helping in overcoming the limitations of simulation and speeding up the verification closure. With the tremendous future works identified, adopting and scaling up of this methodology becomes an absolute necessity.

REFERENCES

[1] Emulation – Synopsys: https://www.synopsys.com/verification/emulation/zebu-server.html
[2] HBM3 IP solutions: https://www.synopsys.com/designware-ip/interface-ip/hbm.html
[3] Corelink DMA-330 DMA Controller Technical Reference Manual
[4] Emulation for accelerating the Functional Verification Closure presentation: https://drive.google.com/file/d/1_-PRqnHApA_0jsYgEpX05oPihXkTN6SX/view?usp=drivesdk