A Full-scale System Monitor and Evaluation Solution for SoC Verification

Bin Liu, Modem Hardware, Intel, Xi’an, China (bin.rocker.liu@intel.com)

Haibo Shao, Modem Hardware, Intel, Xi’an, China (haibo.shao@intel.com)

Xiuqin Zhang, ASIC Design Center, UnilC, Xi’an, China (xiuqinx.zhang@intel.com)

Xinpan Man, Modem Hardware, Intel, Xi’an, China (xinpan.man@intel.com)

Modern SoC size and complexity grow too fast, which brings a big challenge to strive for a qualified verification through the module level to the system level. However, the verification strategy between module level and system level could not take the same strategy and evaluation methods. It is due to **the difference of not only the target size for simulation load but also the verification objectives**. For chip level, regarding its billion gates body, we figured out new ways to decrease the simulation load [1], and apply C code for processor programming. At the same time, **chip level would take more focus on the subsystem integration check, integral system health evaluation, network bandwidth calculation, balance between speed and power and so on**.

Legacy advanced verification methodologies themselves are designed for function verification, and thus assigned clear roles to verification component as driver, monitor or scoreboard. When moving to chip level, those module environment would be considered for hierarchical reuse, but existing verification components are not helpful to satisfy the new challenging requirement listed above. From the UVM well-structured feature, it is still **a good choice with UVM to establish a unique system monitor framework for SoC specific verification requirement**.

In the following, it will list the features of system monitor solution, and also shortly explain their implementation basics. Such **a full-scale system monitor and evaluation solution, it has been developed in-house and internally called ‘Hawkeye’ for its great abilities covering specific monitor requirements**. By the abstract submission date, Hawkeye solution has been successfully applied in executing project, and in deed **assisted verifiers in system debug, status identification, coverage completion and also performance improvement.**

The listed Hawkeye features involve:

- **Processor monitor**
- **Register monitor**
- **Unified system coverage**
- **Clock monitor**
- **Performance evaluation**

Processor Monitor

Large SoC in general would have dedicated processors which are allocated in each subsystem to serve central control. Meanwhile, those distributed processors would also be acting in parallel to access targets through the interconnecting network. Although processor IP provider (e.g. ARM, Synopsys) has corresponding solution to do post extraction for processor execution history, it is still not easy to get understood what each processor is doing real-time unless exploring their boundary bus activity and system signals.

The ‘processor monitor’ would keep eyes on dedicated processors for their activities and status:

- The **address, data and register/memory name analysis** for each bus transaction
- The **power, reset and clock**
- **Interrupt** input

Register Monitor

In module level, it is available to build UVM register model for register test and coverage. For chip level, we have collected all of necessary IP UVM register blocks and organize to form a singleton SoC register model. Based on such a SoC register model, we would implement those features:

- Monitoring each register block slave **interface transaction for register coverage sampling**.
- Developing a **visualized tooling for ease read and write register** via the UVM register model backdoor access.

Unified System Coverage

SystemVerilog supports fine-grained coverage definition, but it faced issues when being imported to chip level verification. A big limitation is that the big re-elaboration time consumption does not allow frequently defining the SV coverage. Referring to our successful project experience, we followed **a DPI-C layer to bridge SV coverage definition and C coverage APIs**. **With C APIs, SoC verifiers would easily embed the system verification definition into existing test code**. The available coverage C functions include:

- **single** cover bin for one signal
- **transition** cover bin for one signal
- **range** cover bin for one signal
- **cross** cover bin for two signals

Clock Monitor

Clock is an easy-to-forget topic when composing system test because **verifies used to do clock configuration but easy to forget checking the real clock frequency**. Clock gating and frequency reduction application have been extensively used in system design for power saving, while verifiers still only focusing on function check without cares on the clock check.
'Clock monitor' feature is to **extract a single sourced clock information database with expected clock frequency and signal path, and compiled all clocks with unique names**. When running simulation, **core clocks would be always monitored for system status, and each case would also specify additional clocks for monitor, and check their on/off state and frequency**. This way would be also important when walking through gate level simulation since clock frequency should be absolutely configured correctly unless slow clock speed would be not helpful to identify critical timing corner cases.

Performance Evaluation

Performance evaluation is also a key feature to purse the market and customer demands. For those huge data transmission devices (e.g. DDR, USB, PCIe), it is **necessary to validate the real hardware bandwidth in pre-silicon phase** as early as possible, and also better to leave time for design change when performance is not satisfied. For performance evaluation, we have **bind bus monitor around core data transaction designs, and keep dynamic calculation along several massive data transmission paths**. The evaluation is based on point-to-point data paths which would cover sensitive data paths' bandwidth.
With recorded data transaction statistics, it is available to analyze the performance and draw bandwidth curves on-the-fly. During simulation, **verifiers would easily obtain the bandwidth's rise and fall, and issued bottleneck points through the data path**. Then hardware weakness improvement or software modification would be follow-up.

Hawkeye is still an on-the-way solution and we are making it to be **more feasible to solve system verification pain points**. By the paper abstract is submitted, we are also exploring the **possibility and advantage to introduce power evaluation feature**. Quick power estimation topic should be another interesting topic, maybe we would share its draft shape in next year DVCon.

[1] Bin Liu, Haibo Shao, and Hongbin Yu, "Best Practices over Enhancing SoC Verification Efficiency", DVCon China, April 2017.