Synthetic Traffic based SOC Performance Verification Methodology

Jeonggu Lee, Taewon Park, Hyungtae Park, Taeyoung Jeon, Hyunjae Woo,
Youngsik Kim, Seonil Brian Choi
Samsung Electronics Co., LTD. (jeonggu.lee@samsung.com)

I. Abstract

The SOC use-case scenario is becoming more complex, and more than 20 functional IPs (such as CPU/GPU/NPU/CAM/DPU) are interoperating to perform one scenario. If even one IP does not satisfy the performance, the scenario must be modified to downgrade, spec. out or silicon re-spin may occur. Therefore, emulation-based performance verification in pre-silicon stage is becoming increasingly important. For Performance Verification (PV), we perform Synthetic Traffic based (STB)-PV using a traffic generator and Real Traffic based (RTB)-PV to operate real RTL IP. Since the IPs bring-up to perform RTB-PV is prepared just right before the RTL freeze, bugs found in the RTB-PV cannot be fixed on time. So it is necessary to re-spin silicon to fix the bugs. In this paper, we propose STB Stress Test (STB-ST) and STB Bus Performance test (STB-BP) methodology to filter out many bugs at the early stage of project.

II. Introduction

Before we introduce our methodology, we would like to define what SOC Performance Verification (PV) is. SOC PV verifies 1) whether all system IPs are designed with spec performance, 2) whether all main bus's traffic going through the memory path meet the target performance, such as bandwidth (BW), Multiple Outstanding (MO) and latency, and 3) whether the SOC meets the customer's KPI performance, such as Benchmarks score and frame rate of the worst case scenario.

SOC PV proceeds to the following three steps in consideration of the project schedule and RTL readiness.

At the beginning of project, the main system buses and memory subsystem may only available to run. The system bus and memory subsystem are major system backbone IPs that determine the overall performance of the SOC and need to be verified in the early stage. The STB-ST verifies the system backbone performance by creating various random combination scenarios using Traffic Generators (TGs). Once IP integration tests are passed, we start STB-BP to verify internal system IP such as MMU, security, up/down sizer and buses. Finally, functional IPs are ready to enable, we start to generate all IPs vector for each scenarios and all IPs corresponding to the scenario are interoperating to RTB-PV. Because of project schedule, RTB-PV cannot cover all real use case scenarios, so only several worst-case scenarios are verified. Some bugs are found at this stage, and silicon re-spin may be required because there is not enough time to fix them. Therefore, it is very important to find bugs in the early stages and increase verification coverage, so this paper propose STB methodology and key components to implement.

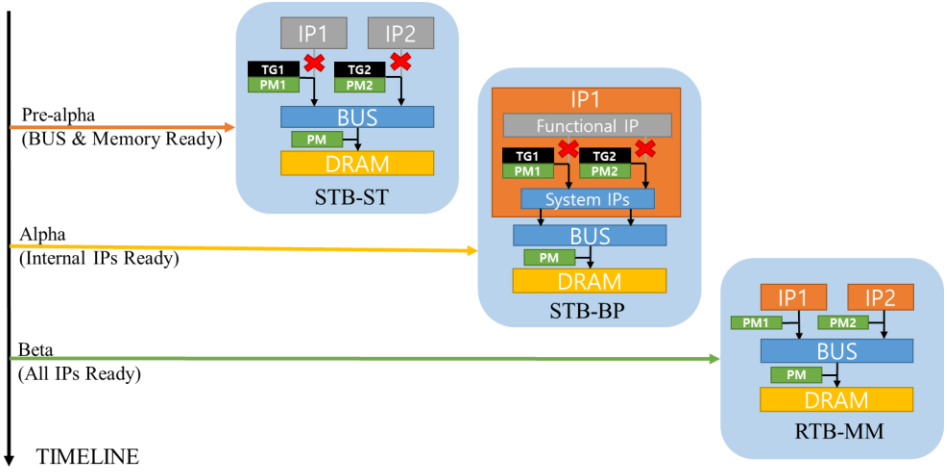


Figure 1. Performance Verification Flow

III. Related Research (Previous work)

This chapter introduces the reasons and problems that RTB-PV must be performed through related research. RTB-PV configures the sensor, display module environment, and vectors of functional IP using RTL in the same way as in the actual silicon, and allows the same traffic as the silicon operation to be generated as much as possible. For example, if you look at the Multimedia scenario below, it shows the process from Image Sensor to Camera sub-system, Video sub-system, Audio sub-system, and Display sub-system to Display Interface.

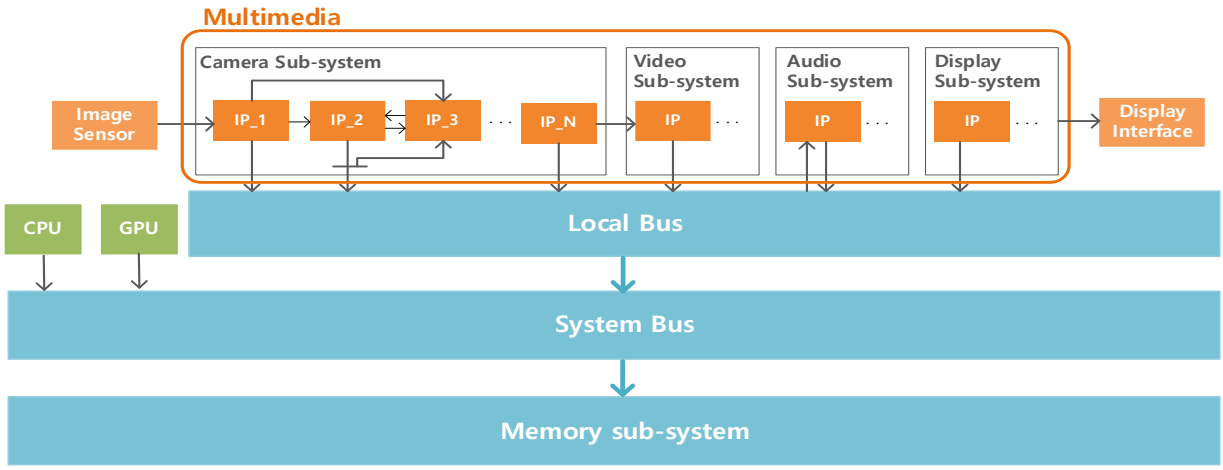


Figure 2. Example of Multimedia Scenario

The most powerful advantage of RTB-PV is that it is real traffic. It is a great advantage to be able to preview the same performance as silicon before silicon comes out. Therefore, it is one of the indispensable processes. In order to do RTB-PV, it is necessary to verify the operation of all other IPs corresponding to the scenario as well as image sensors or display interface modules. In addition, many of these IPs are interlocked with each other. RTB-PV can be started when RTLs are matured and it takes more than one month to prepare scenarios for more than 20 IPs. Therefore, some performance bugs cannot be fixed in time, and even a silicon re-spin is required. For performance verification, the main bus topology and memory subsystems must be stabilized and verified at the beginning of the project. This is because if the bus topology and memory subsystem are changed, many other IPs are also affected. As we shared before, RTB-PV cannot generate many test scenarios to cover all corner case on time.

To break through these problems, we tried to develop Synthetic Traffic based new methodology that can verify performance earlier and cover more corner cases. At the end of 2018, STB was first brought up, and at that time, only dram utilization was measured, and the overall operation was very slow because the TG operation increase wall-clock time. Therefore, it was necessary to make STB more powerful by TG enhancement that can simulate real traffic and by improving PV workflow.

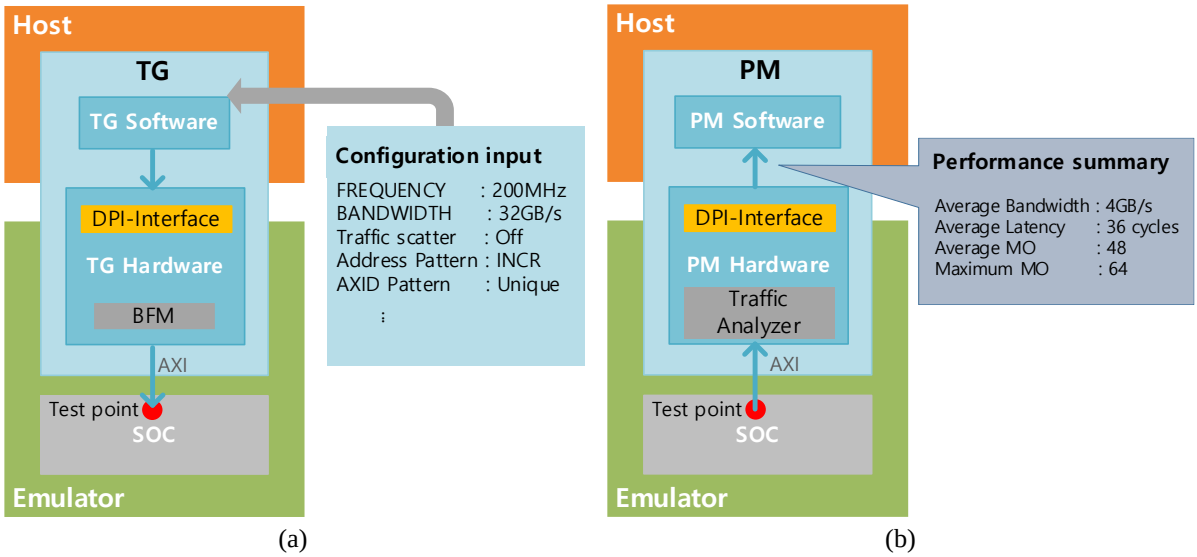
IV. Proposed Methodology

This chapter introduces the STB-PV methodology. The proposed methodology consists of key components, verification automation and features of STB-ST/BP. To perform performance verification at the early stage of the project in time, it is crucial to use Traffic Generator/Performance Monitor (TG/PM) for STB-PV, and automation is necessary. The following sub-sections explain more detail about features of the STB-PV, and explain how to verify the performance of the SOC in the early stage of the project with reducing human error.

A. TG/PM for Synthetic Traffic based (STB)-PV: Before explaining the proposed TG/PM, we would like to show you what important things to consider when implementing. The connection between emulator and host Linux machine (Run time host) are connected by PCIe (Peripheral Component Interconnect Express). Therefore, the bandwidth is quite huge enough and it is up to 32 GB/s as current emulator, so the bandwidth is not a bottleneck nowadays, but the problem is latency overhead. According to the PCIe specification, 4 GB data read transaction only takes less than one second. However, it may take more than five seconds in the real emulation, depending on various conditions. One way 4 Byte transaction takes 5 us ~ 7 us on PCIe specification, but it may take 14 us at the best on real. Let us have extreme situation. If the transactor is supposed to read 4 GB from Host Linux and it sends 4 Byte read transaction for 1 G times, it will take over 7 hours. But if the transactor sends 64 Byte read transaction, it

will take only around 10 minutes [5]. Therefore, it is necessary to reduce communication counts between emulator and host Linux machine as less as possible to enhance emulation speed. To do that, we use the HW buffer to store the multiple transactions and transmit these at once. In addition, instead of the raw data, HW calculate and process the raw data to result and transmit the result only.

To meet these requirements, we proposed improved TG/PM for STB-PV. Figure 3 shows the architecture of the proposed TG/PM. The TG/PM has two parts, the SW part and the HW part. The SW part resides on the host side and takes the user's input or gives output to the user. A SW part is required for flexible TG setup and collecting performance summary. The HW part resides on the emulator side. In case of TG, it generates synthetic traffic, which TG puts into the test point, and in case of PM, it calculates and summarizes performance results. In this architecture, SW and HW must communicate with each other, so TG/PM has a DPI interface. As described above, it is important to reduce the number of transactions while communicating on both sides, so there is a buffer in the HW part to reduce the number of transactions and maintain the emulation speed.



(a) (b)

Figure 3. Traffic Generator and Performance Monitor for STB-PV

One of the core enhancement functions of TG is to support various traffic patterns. The actual master IP is replaced by a TG, which mimics the traffic pattern of each master IP in the SOC. As shown in Figure 3-(a), AMBA AXI protocol-based TGs can generate a variety of traffic according to user settings such as frequency, target BW, burst length, address pattern, etc. The existing TGs only generate average BW traffic as shown in Figure 4-(a) [1], but it is important to support various traffic patterns to increase accuracy as shown in Figures 4-(b), (c), and (d). In our experiment, it is insufficient to use general average BW traffic, because performance test using such a traffic pattern results in no performance problems, but performance test using the traffic pattern as shown in Figure 4-(b) could result in performance degradation of SOC, which is a performance bug. Therefore, supporting various traffic patterns to increase the coverage of performance verification is one of the important features in the proposed TG. Another improved feature of TG is to support replay. Some specific IPs, such as MMU and LLC, are sensitive to address patterns, and using specific patterns in TG settings is not enough or even meaningless for performance verification. To overcome this, the proposed TG can generate prepared traffic traces containing address patterns of actual traffic, enabling more realistic performance verification.

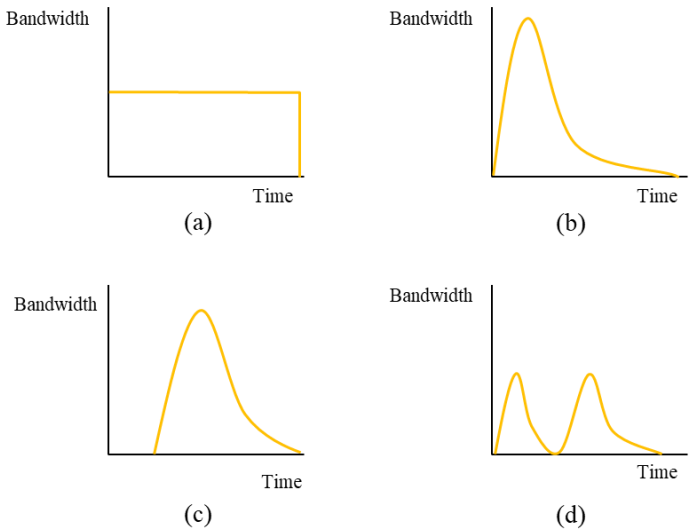


Figure 4. Various Traffic Patterns

PM is a component, which gathers performance data at each test point [2]. PM gathers and displays a performance values between bus and memory controller. To save wall clock time, HW PM calculates each PV values in emulation time and only final performance values calculated in emulator are transferred to the host machine through DPI (Direct Programming Interface) call, and then, the values are written in file. To reduce the loss of the emulation speed due to these DPI call, The PM calculates and transmits the result in frame units (e.g., 1 ms). As shown in Figure 3-(b), PM transmit the results of BW, MO, and latency in frame units. Using the average, minimum, and maximum results in frame units is sufficient to analyze the performance trend of the SOC and does not degrade the emulation speed.

B. SM (Scenario Manager) Implementation: SM is developed by using synthesizable SV (System Verilog) and DPI interface. It consists of two components, an SM parser and an SM controller as shown in Figure 5. An SM parser configures scenario files based on user input constraints, which are usually PV parameters. A user constraint is determined by selecting AMBA AXI bus parameters that affect performance. The values for BW, address patterns, burst lengths/sizes and Qualify of Service (QoS) are selected. Based on selected criteria, a scenario file, which contains scenario descriptors is automatically generated. Each scenario descriptor has information that is inputted to a SM controller. An SM controller manages all TGs and PMs based on the automatically generated scenario descriptors, which configure TG for each master. TG loads computed traffic information that is previously generated by an SM parser and generates traffic patterns accordingly. An SM controller controls TGs by generating command signals to TGs while monitoring the status of each TG. The controller shares verification scenario information in a descriptor such as an expected verification runtime and a scenario index that identify a certain scenario among all scenarios to a PM module. Each PM stores performance information separately based on scenario indices. An SM controller launches all generated verification scenarios and at last, an emulation session is closed.

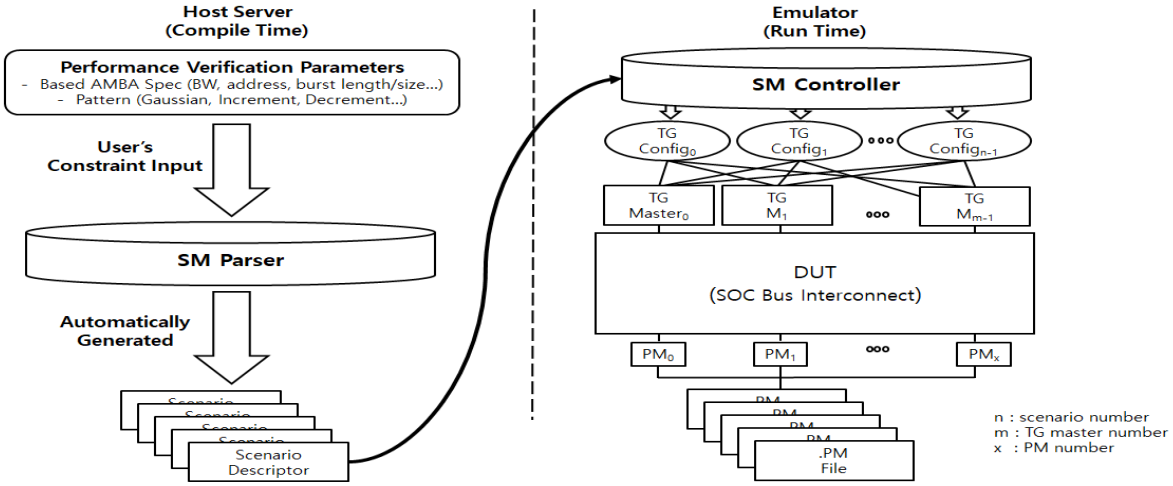


Figure 5. Scenario Manager for PV

All functional IPs are connected to their own system IPs such as MMU, security, up/down sizer, address re-mapper and so on. These system IPs can block the performance of functional IPs due to insufficient MO, buffer, BW, and long latency. Because those performance bugs are hard to fix by function and metal ECO, all system IPs should be verify performance at this early stage in STB-BP. To verify this, we replace functional IP by TG at the AXI master port and verify that system IPs are designed to meet the performance requirements such as min/average/peak latency, max MO and max BW.

C. IP-XACT for automation: By the fact that, STB-ST and STB-BP handle hundreds of verification targets, TG attachment for each verification point requires a lot of time and effort. It is needless to say about an automation methodology which can place the TG for every verification points that we want to verify [3]. When a list of verification targets were given, the automation methodology searches the verification point from the design, then generates the TG database file for the emulation image build. The TG database file contains detailed information about each TG. So, it is possible to attach the TG conveniently. Moreover the automation methodology can to its job in a short time. Because it searches the design meta-data, which is provides by the IP-XACT. Compared to the huge design, IP-XACT has small amount of data, so the methodology does not require a lot of time for parsing the signal information for the TG. As a result, the automation methodology can enhance the productivity. Figure 6 shows the input and output of the automation methodology. First, it reads the list of verification target, which has the TG attachment point. Second, it searches the TG attachment point and obtain the detailed signal information from the design meta-data which is provided by the IP-XACT. Lastly, the automation methodology generates the TG database file which describes the TG attachment information. This database file will be used for the emulation image build.

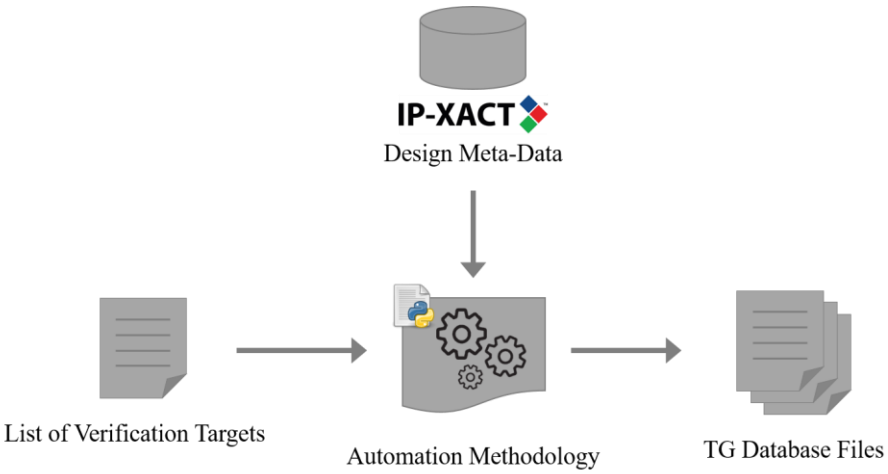


Figure 6 Flow of the Automation Methodology

D. Synthetic Traffic based-PV (STB-PV): Through the above automation step, environmental setup using synthetic traffic becomes possible and various experiments become possible. Tests using synthetic traffic depending on the measurement DUT and purpose can be divided into two PV mille stones, STB-ST and STB-BP.

In the STB-ST stage, the performance of the Network-on-Chip (NOC) and memory controller corresponding to the backbone of the SOC is verified. These blocks determine the main performance of the SOC, and perform performance verification on the backbone RTL implemented at a relatively early time. Verify various combinations of paths connected to the backbone bus, and perform performance tests using various combinations such as traffic pattern and BW. At this stage, it is necessary to check whether the specification of the SOC product satisfies the required performance, and test the BW including the target BW and non-real time IP of BW, latency, dram utilization, and real time IP as a whole. Figure 7 shows the diagram when STB-ST is performed in the SOC bus. As introduced above, you can see that the TG that can generate the synthetic traffic and the PM that can measure the performance in each path are connected. In the NOC bus corresponding to the interconnect of the bus, several TGs create various traffic patterns corresponding to the scenario. At this time, the display IP and camera IP corresponding to the real time IP must be guaranteed real time performance. If performance is not guaranteed, the screen may be disconnected or the frame may be omitted during camcording. To ensure these things, there is a QoS that can adjust the traffic of the SOC bus, and you can check if this QoS operation operates according to the spec. The priority to guarantee the BW of the IP corresponding to the real time can be adjusted. If the service of a specific IP is delayed, an urgent event occurs to increase the priority.

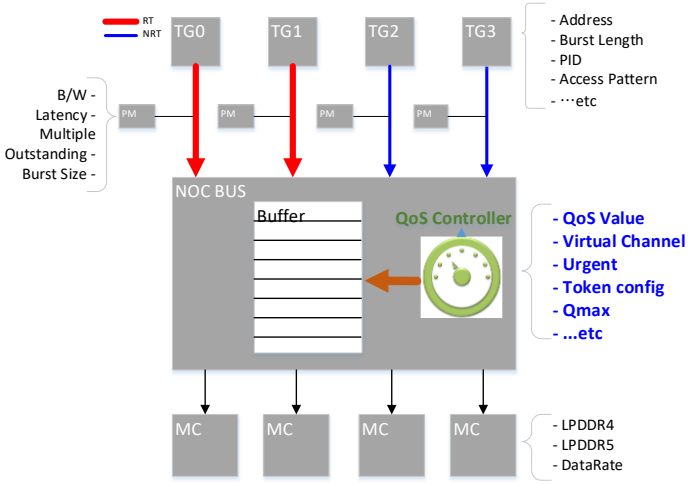


Figure 7 Diagram of STB-ST

In the STB-BP stage, the test is performed including system IPs at the block level following the backbone of the SOC. Since system IPs are included in time, it can start slower than the STB-ST stage, but the coverage of the DUT is higher because the block level is included. STB-BP includes SysMMU, up/down sizer, and local bus that may affect each block performance. During STB-BP test, we check whether target BW is measured for each path, whether latency requirement is satisfied, and whether MO of AXI bus is fully digested.

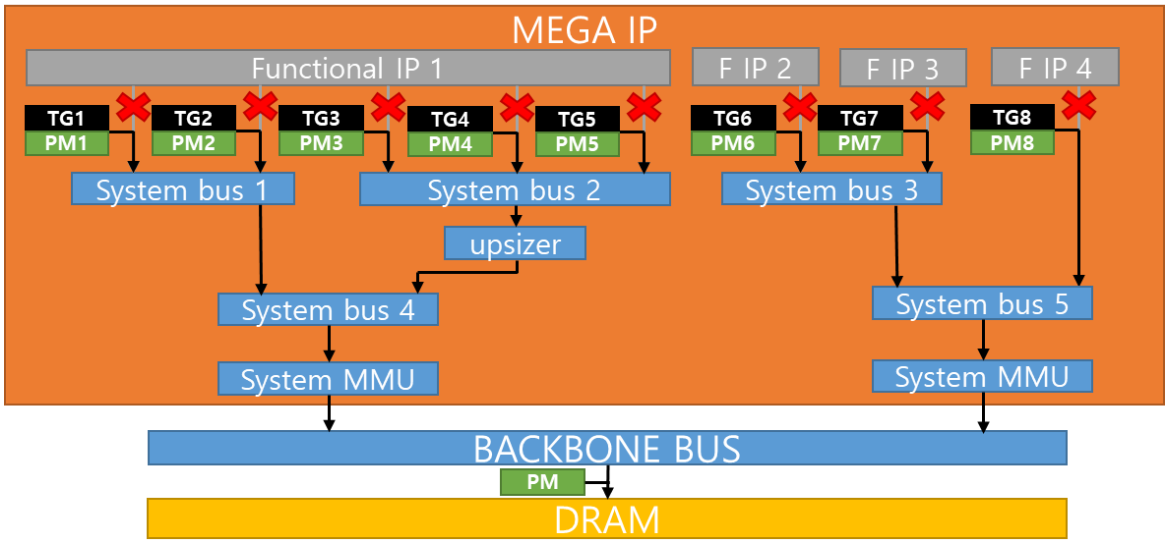


Figure 8 Diagram of STB-BP

In the MEGA IP example as shown in Figure 8, each master port goes through many internal buses, MMUs, US/DS, etc. until it reaches the backbone bus. If TG/PM is attached to each master port by applying STB-BP, the path to DRAM can be scanned as a whole, so that internal buses, MMUs, up/down sizer, etc. can be verified as well.

Additionally, TG's address patterns are usually set to be general, but if the replay function mentioned previously is used to simulate IPs with special address patterns, special traffic patterns and address patterns can be reproduced, so MMU performance can be evaluated through this using hit rate.

V. RESULT

With the introduction of STB-PV methodology as above, it has two main effects. First, the very precise test result became possible. The result of DUT can be directly compared to the specification of the result of model within an error of less than 5% in performance metrics like BW, latency, and MO numbers as shown in Figure 9. Some bugs we found were the ARQ buffer size bug and the long latency issue caused by clock gating. These were detected by comparing the result of RTL and Model which was not previously easy [4].

IP_Name	RTL (MB/s)	Model (MB/s)	RTL/Model (%)	Pass/Fail	Note
_D0_C	12800	12797	100.0%	Pass	
_D1_C	12800	12797	100.0%	Pass	
_D2_C	12800	12797	100.0%	Pass	
_D3_C	12800	12797	100.0%	Pass	
_D4_C	12799	12797	100.0%	Pass	
_D_C	12799	12797	100.0%	Pass	
_D0_D	25600	25596	100.0%	Pass	
_D1_D	25600	25596	100.0%	Pass	
_D0_M	10500	12799	82.0%	Fail	ARQ buffer bug
_D_A	10500	12799	82.0%	Fail	ARQ buffer bug
_D_B	12800	12797	100.0%	Pass	
_D0_M	12799	12797	100.0%	Pass	
_D1_M	12800	12797	100.0%	Pass	
_D2_M	9120	9347	97.6%	Pass	
_D3_M	655	669	97.8%	Pass	
_D4_M	655	669	97.8%	Pass	
_D5_M	654	669	97.7%	Pass	
_D0_R	12800	12797	100.0%	Pass	
_D1_R	12799	12797	100.0%	Pass	

Figure 9 Result of STB-ST

Secondly, PV coverage can be increased due to effective execution as shown in Figure 10. IP-XACT and TG automation enable seamless processes from architecture to RTL development, which can increase test efficiency.

Human error decreased in TG insertion and configuration. The easy TG configuration reduced the time required to configure the scenario. Therefore, in the early stage of SOC RTL design, more test scenarios can be executed and performance can be checked before RTL freezing [4].

	Item	Previous (Manual conversion)	Current (IP-XACT & Automation)	Difference
Efficiency	TG Automation	8 hour	10 min	▼98%
	Sceanrio Genration with TG	8 hour	10 min	▼98%
	Total time	16 hour	20 min	**▼98%**
Coverage	**Sceanrio(#)**	50 (Max.)	4150	**▲83 times**

Figure 10 Increased PV Coverage

VI. CONCLUSION

The proposed PV methodology was applied to the flagship mobile AP SOC project. More than 2000 scenario experiments have been completed by STB-ST and STB-BP before RTL freezing and almost all system performance bugs have been fixed before RTB-PV begins. The automation flow make it possible to insert 400 TG/PM for building emulation platform and run and analysis 2000 scenario right on time. We have increased PV coverage using STB scenario and will enhance TG to generate more realistic traffic for mimicking real scenarios in the future.

REFERENCES

[1] SeungHyun Song, “Synthetic Traffic based Performance Verification.” In Proc. DAC, LV, USA, June 2019.
[2] Hyungtae Park, “SOC System Bus Performance Monitor (SPM).” In Proc. DAC, LV, USA, June 2019.
[3] Taeyoung Jeon, et al. “Automation Methodology for Bus Performance Verification using IP-XACT” DVCon, USA, 2023.
[4] Hyungtae Park, “Sophisticate Performance Verification Of SOC BUS By IP-XACT And Model”, DAC, SF, USA, July 2023.
[5] Hyunjae Woo, “OS-aware IP Development Methodology”, DVCon USA, 2019.