2022 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE
MUNICH, GERMANY
DECEMBER 6 - 7, 2022

A novel approach to standardize reusable Modular Plug and Play Skeleton Structure (MPPSS) to expedite verification closure

Himanshu Dixit, Senior Staff Engineer , Samsung Semiconductors India Research, Bangalore , India (hd.dixit@samsung.com)

Chandrachud Murali, Associate Staff Engineer, Samsung Semiconductors India Research, Bangalore , India (chandru.m@samsung.com)

Sriram Kazhiyur Soundarrajan, Associate Director, Samsung Semiconductors India Research, Bangalore , India (sriram.k.s @samsung.com)

Somasunder Kattepura Sreenath, Director, Samsung Semiconductors India Research, Bangalore , India (soma.ks@samsung.com)

Abstract— With time to market and first pass silicon being crucial factors defining the success of an organization, challenges have become manifold in delivering good quality DV deliverables. System on Chip (SoC) designs have become more challenging with complex functions implemented on hardware with increased complexity in IPs. With this increased complexity, the runtime of the simulation has increased manifold. With time to market being a significant factor, delays arising from run times not only requires budgeting but also needs to be mitigated by out of the box thinking and solutions, beginning from the planning phase. The paper explains the strategy used for verifying MIPI CSI at different abstraction levels with reusable architectural components by Modular Plug and Play Skeleton Structure (MPPSS) that can be leveraged across IPs. This "divide and conquer" method has been in existence for quite long but without a standardization of architecture, approach and different scopes targeted at different abstraction levels. The novelty of MPPSS lies in intelligently pre-visualizing the modular reusable components, subsystem smoke test sets based on design implementation, planning, reuse and sim speed optimization resulting in early bug detection and scenario optimization. The MPPSS architecture resulted in optimizing scenarios by ~91% at SoC and ~82% at the sub-system level for flow flush, ~85% early bug detection apart from saving 5.5x (~70000 hours) on sim time and around $27000 of license cost.

Camera system is data intensive in terms of both volume and compute due to performance and quality requirements. The scenarios required for verification depends on the formats, lanes supported, format conversion, upscaling and downscaling features that are supported by the system. The runtime benchmarking from previous projects led us to an anticipation of a humungous runtime (approx. 12-16 hrs.) in RTL simulation. The simulation numbers revealed that having a separate subsystem and SoC environment will not be sufficient to thoroughly close the verification activity on time with due quality. The need for a tight coupling between SoC and sub-system verification beyond stand-alone closure resulted in the SoC team owning the sub-system. Basic data path, core features, clocking, power scheme tests, GPIO connection, interrupt tests and toggle coverage requires the tests to be run at SoC level of abstraction. Modular standalone blocks, which are specific to IPs in the sub-system, were encapsulated for scaling at different level of abstraction. The wrappers built around the encapsulated verification components resulted in direct reuse of the sequence, interfaces and monitors (Figure 1).

The sub-system comprises of reusable / non-reusable multiple modular stand-alone blocks.

1. Reusable blocks
    a. Third party or vendor procured blocks (C/DPHY VIPs and AXI VIPs)
    b. In-house developed blocks - interfaces, monitors, sequences
    c. DUT blocks – PHY, Controller, Image processor, soft logic within the block
2. Non-reusable blocks

a. Static signal drivers

b. Soft logic specific to SoC

Initial estimates based on the partition of the environment and taking into consideration the different reusable modules at different abstraction levels, the effort required to complete the SoC verification was estimated to be at least 75% lower in comparison with full closure at SoC abstraction.

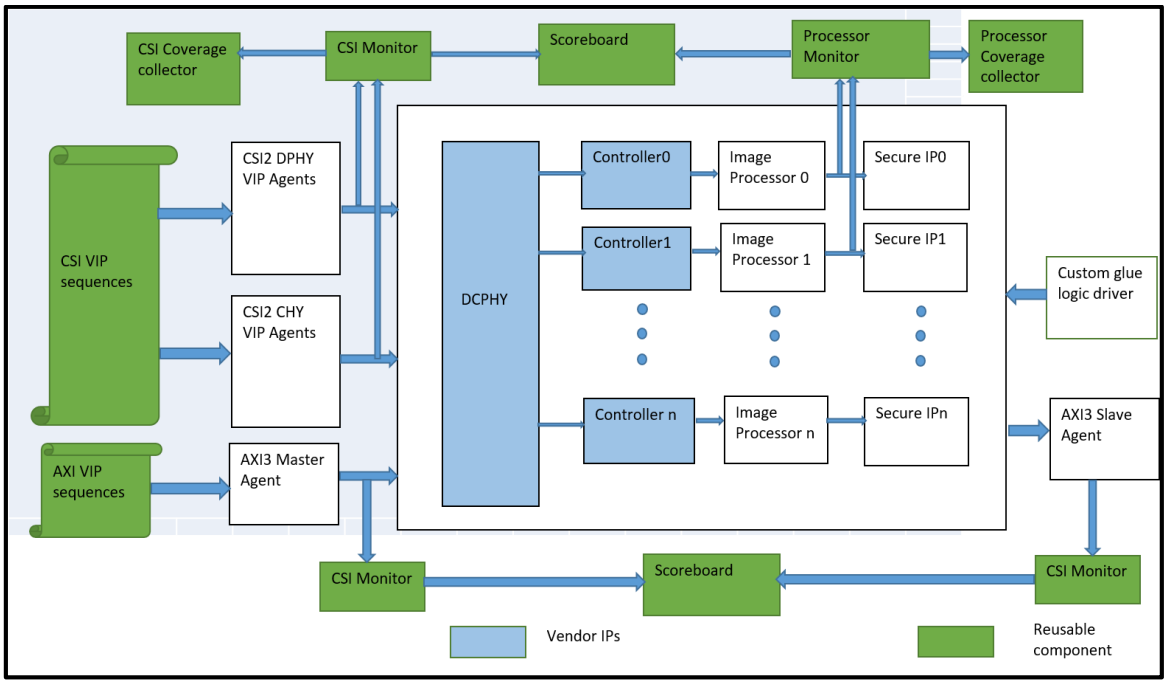

Figure 1. Subsystem Test bench Environment

The formats supported by the system included RAW8 and RAW10. The resolutions supported were RAW8 - 512x512 and 640x640, RAW10 - 1280x720, 16x1080, 16x512, 1920x1080, 1920x16, 512x16, 512x512 and 640x640. The 2 format 10 resolution system supporting different lane, speed and PHY combinations resulted in compounding the verification effort resulting in a need to cover 46000 scenarios. The MIPI CSI Protocol has numerous configuration parameters like virtual channel, different data formats, number of lanes, different resolutions et c. The features were classified into 2 groups image properties – that includes the use case format and resolutions hardware properties that has different hardware configurations applicable for the image sensing operation. Considering the most used lane multiplexing configurations and PHY configurations, a subset of the total combinations was identified as the smoke test set to ensure the working of the most applicable use cases.

There was an another need to create a subsystem smoke suite considering the nature of the design, the volume of test combinations and the very fact that the co-operation of the different components was being tested for the first time. Table 1 describes the different possible scenarios for each kind of image sent along the Camera sub-system. The smoke suite, in turn became the qualification criteria for each successive releases resulting in providing faster feedback on use case issues and confidence for the SoC level verification engineers to proceed with their testing. With thorough analysis of use case and design implementation, the smoke suite of the sub-system was further trimmed to cater to the SoC abstraction qualification requirements. The smoke suites in conjunction resulted in qualifying design at different abstraction levels and saving precious time of the verification engineers. The approach had a lesson learnt on the optimization of scenarios considering that 2 bugs were not detected by smoke, resulting in updating the smoke suite at a later stage.

No. of active lanes	*PHY type*	*Speed of transfer*	*Lane multiplexing*	*Total combinations of the scenarios possible*	*Smoke test set at the Sub-system level*	*Percentage decrease in scenarios (smoke)*	*Total number of scenarios run at SoC level*	*Percentage decrease in scenarios (SoC)*	*Bugs found in sub-system*
4	DPHY	4 slow lanes	Random	240	42	82.5	18	92.5	3
		2 slow lanes + 1 fast lane	Straight ahead	60	11	81.7	2	96.7	0
		2 fast lanes	Straight ahead	10	2	80	1	90	1
	CPHY	4 slow lanes	Random	240	52	78.3	22	90.8	4
		2 slow lanes + 1 fast lane	Straight ahead	60	12	80	2	96.7	2
		2 fast lanes	Straight ahead	10	2	80	1	90	1
3	DPHY	3 slow lanes	Random	960	142	85.2	82	91.4	2
		1 slow lane + 1 fast lane	Straight ahead	120	28	76.7	2	98.3	0
	CPHY	3 slow lanes	Random	960	145	84.9	85	91.1	3
		1 slow lanes + 1 fast lane	Straight ahead	120	31	74.1	2	98.3	1
2	DPHY	2 slow lanes	Random	720	132	81.7	62	91.4	2
		1 fast lane	Straight ahead	40	12	70	6	85	0
	CPHY	2 slow lanes	Random	720	138	81.7	65	90.9	3
		1 fast lane	Straight ahead	40	14	65	6	85	0
1	DPHY	1 slow lane	Random	160	18	88.7	8	95	2
	CPHY	1 slow lane	Random	160	21	86.8	12	92.5	2
Total				4620	802	82.6	376	91.8	26

Table 1. Design related scenario combinations

While developing the sequence of programming for the different IPs, the sub system acted as the benchmark for qualification of the sequence to be used at SoC level abstraction. The sequence development had multiple challenges arising due to iterative corrections of programming guide, design changes and customer requirement changes. The golden sequence qualified by subsystem abstraction was readily plugged at SoC abstraction resulting in saving enormous run times during the iterative golden sequence development phase for all data paths. The concept was widely appreciated by all stakeholders considering the fact that the data path bring up was completed in a span of 3 days which otherwise would have taken 2 – 3 weeks. A total of 14 critical integration issues of the IPs within the sub-system were found in a span of less than 1 month. 12 more issues were caught in the next few weeks resulting in ZERO ECOs for the project.

The lower level abstraction verification ensured that all integration bugs between the functional IPs are identified and fixed. At SoC abstraction, the challenge arises due to soft macro inclusions, which cater to power/clock/interrupt and other management units. The closure of verification at lower abstraction resulted in ensuring ZERO functional issues of the subsystem at SoC. A total of 5 bugs were identified at SoC abstraction arising due to soft macro implementation. The lower level abstraction was utilized to clean up zero-time simulation as well for the first time resulting in early flow flush for the verification changes arising due to netlist environment. Coverage sign off typically used to have 17 - 23 iterations in SoC regression which eventually requires ~1 - 2 months to exercise the intended scenario and close in the merged coverage. However, the reusable component and sub-system for DV code coverage facilitated the coverage closure within 5-6 iterations of regression.

Simulation Flavor (Simulation time in minutes)	*Test category*	*Reusable from sub-system to SoC? (Y/N)*	*Average SoC simulation time*	*Average subsystem simulation time*	*Improvement factor*	*Total SoC Simulation time per iteration (without SS)*	*Total SoC Simulation time per iteration (with SS)*	*Percentage reduction in SoC simulation time*
RTL	4 sensor parallel	Y	390	18	21.67	46800	12120	74.1
	2 sensor parallel	Y	480	21	22.86	163200	76800	52.9
	Register access	N	240	9	26.67	2160	2160	0
	Clock gating	N	210	7	30	840	840	0
	Interrupt Scenario	N	330	12	27.5	2640	2640	0
GLS (Unit Delay)	4 sensor parallel	NA	690	21	32.86	8280	8280	NA
	2 sensor parallel	NA	870	23	37.83	29580	29580	NA
	Use case scenario	NA	780	21.5	36.28	7800	7800	NA
POST-GLS (Timing + SDF)	4 sensor parallel	NA	930	NA	NA	11160	11160	NA
	2 sensor parallel	NA	1020	NA	NA	31620	31620	NA

Table 2. SoC simulation vs. Sub-system simulation time

THE MPPSS ARCHITECTURE

The verification effort and simulation time reductions based on CSI verification activity kindled an interest in the quantification of the results. The results were encouraging to take this architecture further as a generic flow that can be reused across different IPs. Based on the fine-tuning, a generic architecture, a Modular Plug and Play Skeleton Structure(MPPSS) was developed with guidelines for verification engineers to leverage this architecture across any High Speed Protocol Interface IPs.

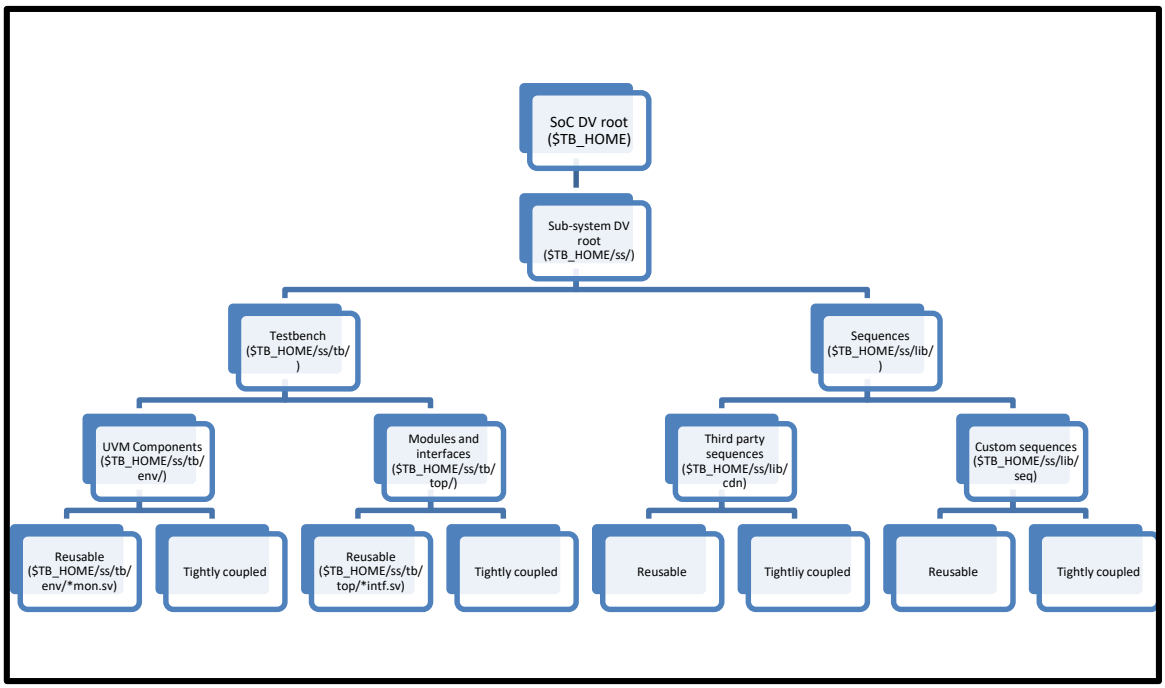

Figure 2. MPPSS Architecture

The structure of MPPSS architecture shown in Figure 2 represents the generic architecture deployed. $TB_HOME represents the top verification directory. The testbench and sequences components are split at first abstraction level to differentiate the root cause of issue during both debug and plug & play. The testbench represents the static component during simulation that includes block configuration, interfaces, monitors etc. The sequence represents the programming sequence and other dynamic SV-UVM objects, which affect the simulation progress. The clear demarcation between static and dynamic components are enabled to flow flush higher abstraction in static environment while lower abstraction makes progress on dynamic environment cleanup. This results in parallel progress at different abstraction levels. The components are further segregated as reusable and non-reusable components to minimize the plug and play porting efforts

Reusability of verification components is a vital process that benefits overall cycle of verification especially when different aspects of verification is covered at different levels of abstraction. The MPPSS based approach compared to a stand-alone and separate verification at SoC and Sub-System is very beneficial in multiple aspects like reduction of simulation time, resources, better controllability, coverage collection, early bug detection, license/slot time saving and data integrity.

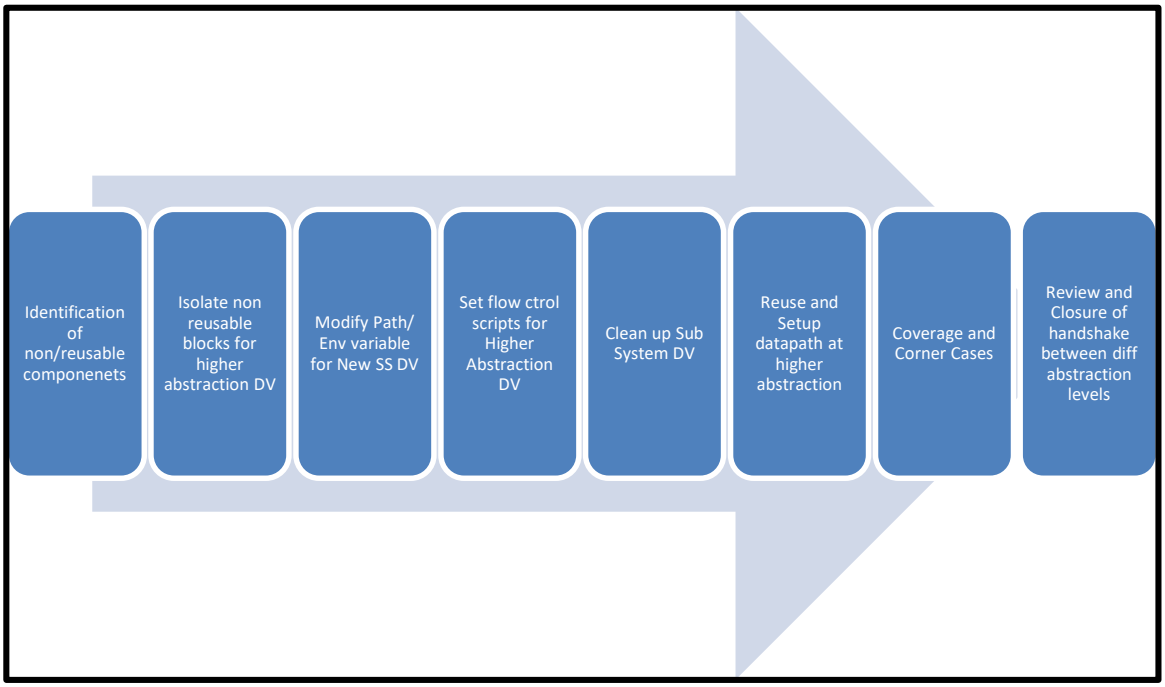

Figure 3. Generic MPPSS development cycle

The architecture has been successfully tested across high-speed interface IPs like PCIe, USB and Display on the ongoing projects. The success of the architecture warranted a wide spread deployment. A through internal audit with verification experts have been carried out on MPPSS architecture. The audit team have recommended enhancements for better error message display management and request for GLS support. Majority of the reviewer's kick started the deployment of the generic architecture across multiple SoC/Sub System DV teams.

REFERENCES

[1] Pavithran, T.M. & Bhakthavatchalu, Ramesh (2017). UVM based testbench architecture for logic sub-system verification. *2017 International Conference on Technological Advancements in Power and Energy (TAP Energy)*, doi: 10.1109/TAPENERGY.2017.8397323

[2] Juan Francesconi, J. Agustin Rodriguez and Pedro M. Julian, "UVM-Based Testbench Architecture for Unit Verification", *2014 Argentine Conference on Micro-Nanoelectronics, Technology and Applications (EAMTA)*, doi: 10.1109/EAMTA.2014.6906085

[3] UVM User Guide 1.2 Accellera, pp. 1-8, October 2015.

[4] MIPI Alliance Specification for D-PHY version 1.2, Sep 2014.

[5] MIPI Alliance Standard for Camera Serial Interface 2 (CSI-2) version 2.0, 7 Dec 2016.

[6] MIPI Alliance Standard for Camera PHY (C-PHY) version 1.1, 7 Oct 2015.

[7] MIPI Alliance Standard for Display PHY (D-PHY) version 2.0, 23 Nov 2015.