Product Life Cycle of an Interconnect Bus: A Portable Stimulus Methodology for Performance Modeling, Design Verification, and Post-Silicon Validation

Gaurav Bhatnagar
Staff Engineer, Analog Devices, Inc
804 Woburn St, Wilmington, MA 01887
Ph: +1(781) 937 2983
gaurav.bhatnagar@analog.com

Courtney Fricano
Staff Engineer, Analog Devices, Inc.
3 Technology Way, Norwood, MA 02062
Ph: +1(781) 461 3962
courtney.fricano@analog.com

***Abstract*-The Portable Stimulus Standard (PSS) is the latest industry standard created to allow the specification of test intent and behavior such that the test stimulus can be re-used across various target platforms. The introduction of PSS alters the traditional approach of verifying and validating a SoC, offering many advantages and a few challenges. This paper explores these process changes with the life cycle of an Interconnect Bus Fabric from System-C based Performance Analysis to verification and validation using a Generic PSS based Traffic Generator.**

I. Introduction

Verification techniques and methodologies have continually evolved ever since the design requirements have started becoming complex. The Portable Stimulus Standard (PSS) is the latest addition to this evolution process and it was created to address the challenge of portability of the tests. The new Portable Stimulus standard allows creation of the test intent such that it can be re-used across different target platforms. Along with portability, the PSS based verification techniques also offers value in terms of visual test representation, constraints setting, dataflow-based randomization and higher test quality. There is a subsequent process change involved in the SoC verification and validation process with the adoption of PSS based techniques and it is important to understand its impact. This paper tries to explore these process changes with a case study of an Interconnect Bus Fabric starting from System-C based performance analysis to verification and validation.

II. Detailed Description

As designs become more and more complex, the process changes like SystemC based modelling, architectural explorations and High-Level Synthesis (HLS) are becoming common along with the traditional design and integration flows. These process changes, in turn, introduce the requirements of checking the adherence to the system design requirements. Different teams involved in these processes have a different kind of platforms and languages to support these platforms. But despite this difference, the basic specifications to each of the successive process is same which leads to a lot of duplication of the effort.

The architectural development team creates virtual platform for architectural exploration and software development using SystemC and TLM based modelling. The component design team designs the Verilog components at block level and integrate them to create a system either manually or with an automated process. The verification at the IP level is typically done using UVM based verification while at the system level a combination of C and UVM based approach is used. The UVM environment allows the checkers to be easily re-used from IP to System level, but the test stimulus is often re-written to work in the top level UVM environment or in C to run on the processor at the chip level. The effort to create tests verifying the startup/configuration and basic mode operation of blocks is repeated for actual silicon testing as new tests are required for the test platform or to run on an evaluation board. Then, the software teams put an effort again to write the software drivers for customer interface.

These duplications of efforts across teams using different languages and techniques, results in frequent false bug reports and significantly delays the overall time to market. A better solution is needed to allow all the test writers throughout the project to speak a common language and allow a large portion of the "simple" functional verification tests to be seamlessly re-used both horizontally and vertically. This different approach is exactly what the PSS based verification techniques brings to the table.

Portable Stimulus Standard (PSS) defines a new test writing language that will allow automatic creation of tests targeted to different platforms from a single test source. In addition to horizontal re-use (simulation, emulation, board level, tester etc.) the new language will allow vertical re-use of tests as well. The tests developed at the IP level would be more easily integrated and reused at the SOC level. The Portable Stimulus works at a higher layer of abstraction that is completely independent from the type of target platform. The target platform here can be a UVM based Verification Environment, a C/C++ based SoC based environment, a C and python-based Post Silicon evaluation platform and so forth.

The PSS based application provide exciting opportunities to create generic applications which can be used to verify the test intent at various levels. One similar opportunity arises when we use an Interconnect Bus Fabric in a Multi-Processor SoC. There are various levels where we need to verify and evaluate the functionality and performance. The interconnect bus fabric needs to be chosen wisely as per the specific requirements of the SoC and hence an early performance analysis is required which can be done using system models (typically in C/System-C). This requires creation of tests which can verify the System Models. Once, the choice of its configuration is made and RTL is generated it needs to be verified at IP level. This requires UVM based verification and UVM sequences to verify the same. Subsequently, the generated RTL is integrated within the SoC System level and it needs to be verified at the SoC level. This is typically done by coding tests in C/C++ for verification and validation.

All these interconnect verification applications can be handled by creating reusable tests using PSS based techniques. To accomplish this, a PSS model was created for a “Generic Traffic Generator” which can create different patterns of Reads and Writes for different number of masters. The Traffic Generator can generate traffic with different distributions to each master so that fast and slow masters can be emulated. Also, it can independently control which master generates traffic and with what frequency as well as create back to back and delayed transactions.

Fig 1 (below) shows the process flow with PSS based traffic generator. The blocks in blue represents the bus interconnect with generic slaves and masters while the blocks in green represents the RTL or Behavioral Models which drives the transactions on the bus. The PSS based traffic generator (represented in pink) integrates and controls these blocks to drive and collect transactions. The traffic generator handles different kind of traffic generation requirements and creates tests for various targets like System-C based applications, UVM and C based tests. Each process needs to be handled in a different way in terms of integration and test generation which will be described in the subsequent sections.

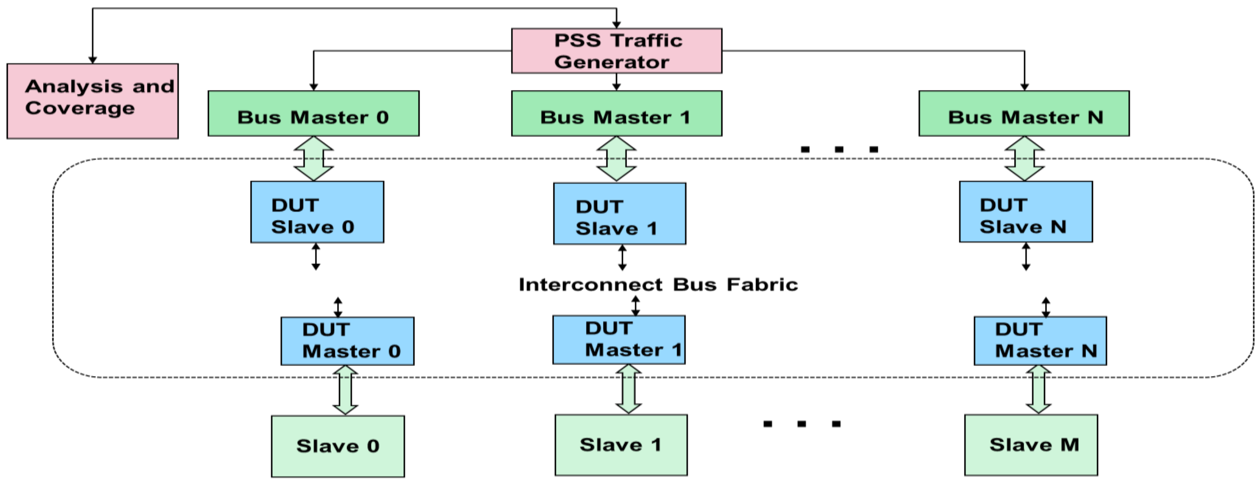


Figure 1. PSS based process flow for Performance Analysis and Verification of Interconnect Bus

A: System-C based Performance Analysis of Interconnect Bus

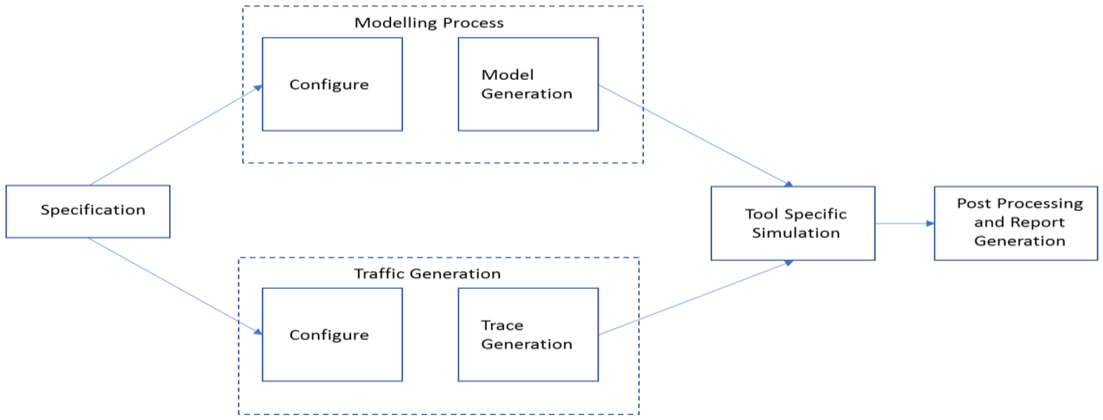


Figure 2. Performance Analysis Flow using SystemC Modelling

The Performance Analysis of Interconnect Bus is carried out to quantitatively measure the system performance and power as early as possible in the SoC Development Cycle. The performance of an interconnect bus needs to be evaluated for a wide range of applications, platforms, and interconnect configurations (topology, features, configuration, etc.). It involves gathering requirements, creating specifications, and then taking those specifications and turning them into a cohesive design that matches requirements for performance, power and area. This is an iterative process which is performed over the course of design. With every iteration, the specifications need to be captured and communicated to design and development teams.

This is achieved with SystemC based representative TLM models of the given SOC specification which can be used to accurately predict the system behavior. Fig 2 (above) shows this flow, starting from configuring the tool for generating the SystemC models. These models are part of the tool kit which can be configured to suit the needs of the design. It can be used to generate cycle accurate or approximate models for AMBA masters and slaves, clock generators, and stimuli. Once the models are in place, the traffic pattern needs to be written at this level either manually, or with directed automated scripts to convert the specification into actual simulation and gather results. The models are then simulated using specific simulators which help provide a solution to quantify the performance.

Despite having modelling and analysis tools, using the tool for more than a handful of candidate designs can be time consuming. Use of scripts to generate the traffic provides certain kind of traffic patterns but exhaustive scenario generation is still a problem. On top of it, as the simulations are time consuming, doing the analysis at the end of the simulation increases the number of trials to achieve the expected numbers. Coupled with the time spent on specification management across the design and the performance modelling functions, we see that there is a need for a more automated flow that starts with a single source. The PSS based techniques are an effective way to address these challenges.

The randomization mechanism in PSS based tools starts from an abstract description of the legal transitions between the high-level states of the DUT, and automatically enumerates the minimum set of tests needed to cover the paths through this state space. The PSS based tools also have a Coverage mechanism which can measure how exhaustively the conditions have been covered in the given state space. This allows measuring the coverage even before any of the generated stimulus have been run thus saving time in the process. The PSS based coverage allows the user to view the transverse paths and allows tests to be generated such that maximum length of the graph is covered. It is thus able to achieve higher coverage in far fewer cycles than the usual constrained random verification.

The PSS based tools also provides a visual representation of the test intent which allows better visualization of the scenarios which can be generated. The directed tests for covering specific test conditions can be easily transverse with this feature. It also allows constraining certain sets of conditions thus creating constrained random scenarios for specific feature set. The introduction of PSS based techniques essentially keeps the flow as given in Fig 2 (above), but the trace generation flow is changed considerably.

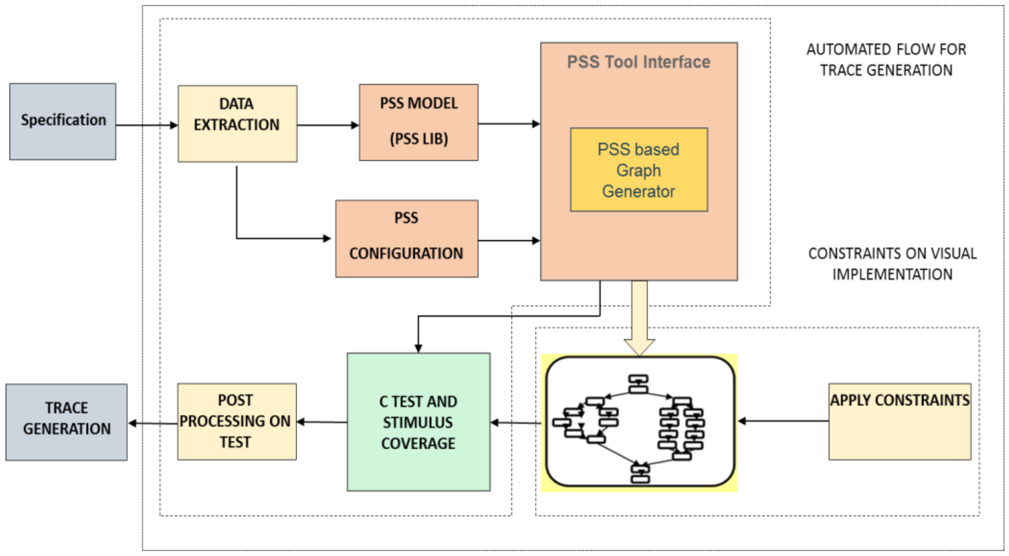
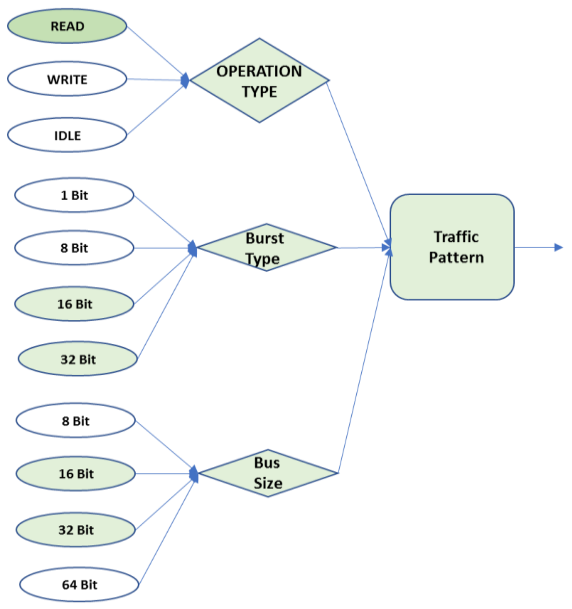


Figure 3. PSS based Flow to generate Traffic Patterns

The heart of the Traffic Generator is a generic PSS Model which contains the algorithm to generate different kinds of traffic patterns. It is a single representation of stimulus and test scenarios. This model can be configured in different ways to generate tests which contains one of the several permutations possible to generate the traffic. It consists of three parts:

- Exec Blocks: The Exec blocks are statements from the external code used in target platforms under the PSS based wrapper. For the System C side application, it has custom code which will perform different types of Reads and Writes to the underlying environment. For the UVM SV part, it has logic derived from tool provided macros (TX Gen) which translates and interact with the SV world with PLI based system calls. It also has a part (C Gen), which can translate and interact with C side and allows seamless reuse across different platforms.
- PSS Model: This is the actual use case model based on entire set of specifications. It consists of collection of functions which will be equivalent of higher level sequences to perform a set of actions. The Traffic Generator contains different set of algorithms which are represented in the form of various simple and compound actions. These functions will eventually call the functions from the Exec Blocks to execute commands on the SV side.
- PSS Configuration: The models being generic needs specific information to generate specific tests. This can be information related to verification like number of AMBA masters, slaves, master type, source and destination address, access type, mean bandwidth, burst size, data size, frequency and bandwidth requirements. This information needs to be derived from the specification to generate a correct representation of the test intent.

Fig 3 (above) represents the PSS based Traffic Pattern generation flow which starts from parsing of the specification. A Python based script parses certain aspects of specification written in the spreadsheets and extracts the data in a certain format which can be read by the generic PSS model and configuration. The PSS Model and configuration are then parsed by the PSS tools to create a Visual representation of the test intent. The Fig 4 shows some part of the Visual Representation of the test intent. It has the conditions represented as write or read operations, single or burst mode, different bus sizes etc. which can be controlled to generate different types of traffic patterns. The part in Green are represented as the conditions which will be transverse while the ones in white which will not be taken into consideration. This can be seen by the user to visualize and constrain different aspects of required traffic. If the user doesn’t add constraints the PSS tools randomly selects certain configuration and create constrained-random tests. This is also a point where tool-based coverage can be collected, and an early analysis of completeness can be made. The tool-based coverage is a measure of how well the test intent has been covered in the tool generated tests. Fig 4 represents parts of the PSS based coverage generated by the PSS tools. The blocks in Red represents the conditions not covered while the Green represents the conditions which have been covered. The user can look at this representation and create the tests for the conditions which are not covered.



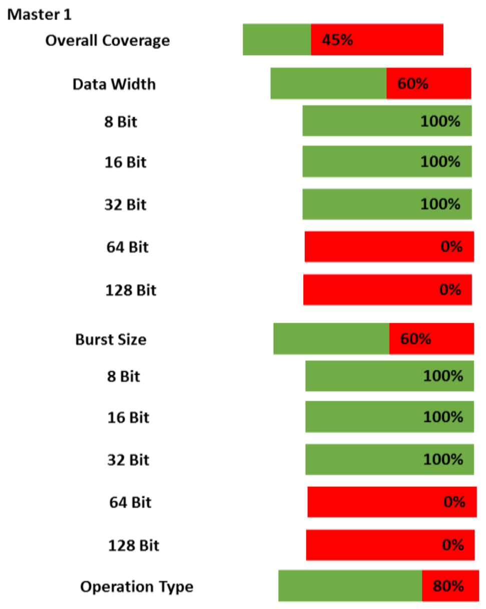


Figure 4. Visual Representation of the Test Intent and PSS based Coverage

Once, the tests are generated a post-processing script is run to create the traffic pattern compatible with the Performance Analysis Simulation Tool in its custom format. The next step is to run the simulations and generate traffic to produce vast amount of raw data which must be processed into different metrics and visualizations to support effective analysis of the results. The Table 1 (below) shows a few examples of the report generated with parameters considered for calculating the Performance Analysis of Interconnect bus for a SoC with multiple masters and slaves. This analysis can have one (master)-to-one (slave) and many-to-one simulations (termed as experiments) to be generated based on a platform specification. Experiments are generated based on a static analysis of the clock frequencies and data widths defined in the platform specification, configured to run up to their theoretical maximum bandwidth. In general, the PSS based traffic allows better placement of the random scenarios targeted for a specific bus configuration. Also, the visual representation of the test intent allows generation of better constraints. The ability to visualize coverage leads to better traffic patterns and thus less number of iterations to reach the highest possible bandwidth in a given master slave system. We have seen an improvement in terms of number of iterations required to reach a highest Mean Simulated Bandwidth with the adoption of PSS based techniques thus saving simulation cycles and analysis time.

By reducing the work required to make architectural performance models of the interconnect as well as a single source of truth in a unified specification, any reconfiguration time is significantly reduced. The flow allows us to explore many design candidates before settling on one to run through timing closure and RTL flows.

Experiment ID	Master	Slave	Direction	Mean Simulated Bandwidth	Mean Static Bandwidth	Mean Simulated Latency
5000	Core	SMMR	Read	1199.72	6000	24
5001	Core	SMMR	Write	999.79	6000	24
5002	Core	L2 Mem	Write	99.92	100	24
5003	Core	L2 Mem	Read	99.92	100	21.34

Table 1. Summary of Data Collected with the PSS Simulation

B. UVM based verification of Interconnect Bus

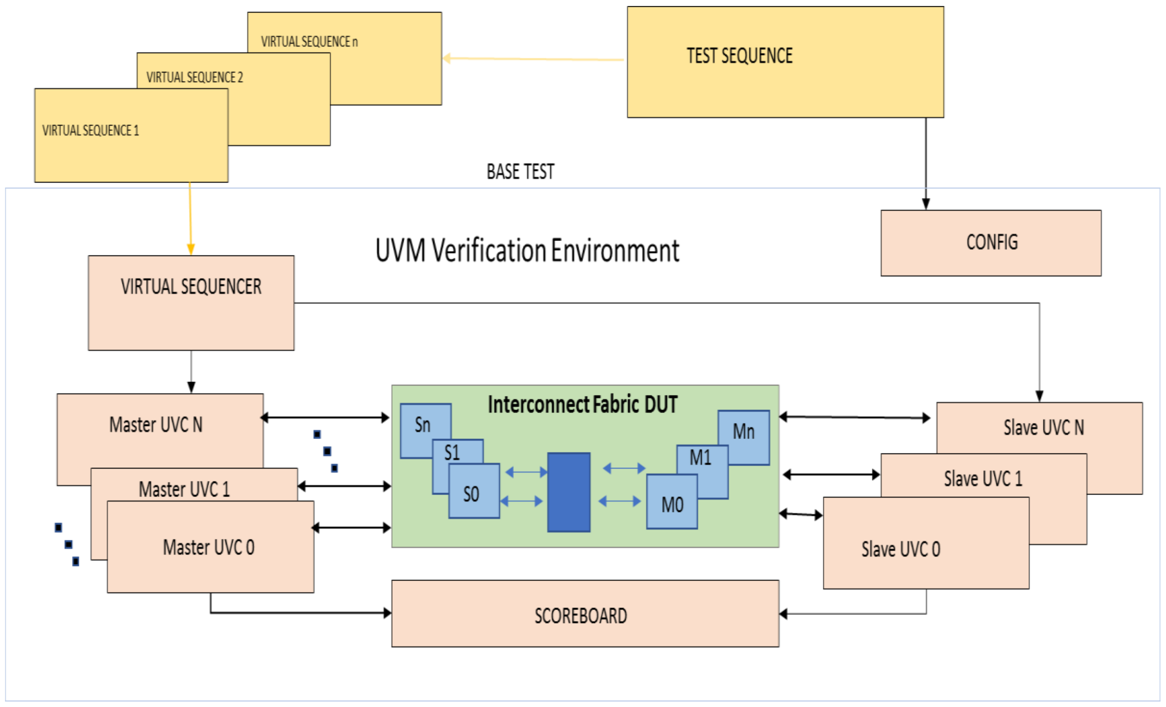


Figure 5: UVM based IP Verification of Interconnect Bus

The Performance Analysis of the Interconnect Fabric gives out the best configuration in terms of performance, power and area. Once, the interconnect configuration is frozen, it can be used to generate AMBA Interconnect RTL with a configurable automated flow. But this flow can be error prone owing to configuration limitations, software structure and manual interpretation of the specifications and thus it needs to be verified to ascertain a flawless generation process. This is achieved traditionally by using the industry standard UVM based approach.

The UVM environment for Verification of Interconnect Bus is shown in Fig 5 (above). It consists of different kind of AMBA (AXI, AHB, APB) master and slave UVC in custom configurations connected to the DUT Slaves and Masters respectively. The environment can be configured with a Generic Configuration for the environment. The Scoreboard records the transactions and indicate error on any kind of data mismatch. The tests contain sets of Sequences and Virtual Sequences which control the functionality of the underlying UVC and interfaces. The tests are run in accordance with the test plan derived from the specifications with the directed and random test cases. The Functional Coverage Points are also created with respect to the verification plan to ensure we meet the specification. After this, the simulations are run, and coverage database is created collecting Code and Functional Coverage. The database is analyzed, and coverage holes are reviewed. The regressions are run, and reports are generated and analyzed. The process is repeated until the desired Coverage goals are reached to ensure high quality verification.

Along with the directed tests as a part of Verification Plan, the UVM based technique relies upon the random tests to achieve the coverage goals. It starts with random stimulus and gradually tightens the constraints until coverage goals are met, relying on the brute power of randomization and compute server farms to cover the state space. While the code coverage is the quantitative measure, the functional coverage is the qualitative measure of the DUT code execution. Typically, this quality is limited by the diligence and thoroughness of the humans who draw up the verification plan and analyze the coverage reports. The other factor which decides the quality of verification is the effective automated checking. A combination of packet comparison using the scoreboards and assertions-based check points can decide the number of Post-Silicon bugs discovered later in the flow. The UVM based verification technique is a self-sufficient and effective in ensuring high quality verification. However, the introduction of PSS based techniques further improve the Verification Flow with its various features.

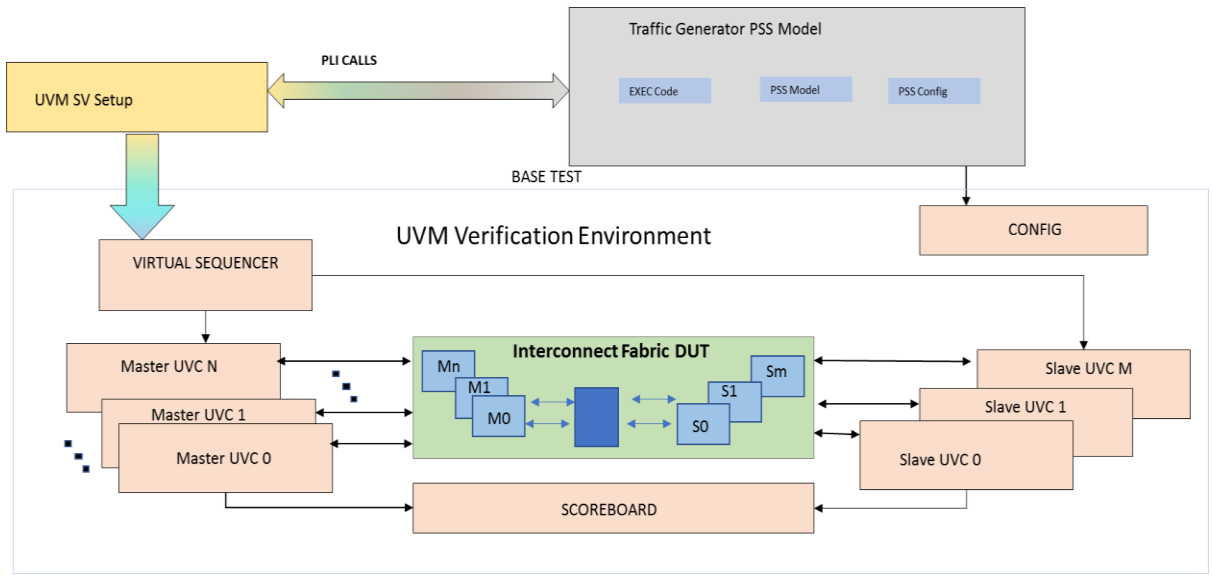


Figure 6: PSS based IP Verification of Interconnect Bus

The PSS based verification starts with the creation of Verification Plan as per design specification and the setting up of the verification environment. The test intent is captured in terms of Portable Stimulus Models, Constraints and Configuration files. The tools supporting this standard can then generate the tests for a given type of Verification Environment and graph-based coverage is collected. The analysis of this type of coverage can indicate the holes in the test constraints and configuration and the process can be revisited.

Fig .6 (above) shows the Verification Flow with the introduction of PSS based models. The important point to note here is that the PSS based Models don't replace the UVM based environment. Instead, it adds on to the existing UVM based environment to enhance its capabilities. The UVM Verification Environment continues to have the Master and Slave UVCs, with the SB and configuration while the Virtual Sequences are bypassed with the UVM SV infrastructure. The environment is controlled by a top level UVM test which on one hand calls the Virtual Sequences to control the UVC operation, on the other hand it interacts with the Portable Stimulus generated format with PLI or DPI based system calls. The PSS Model is completely reused from the SystemC based performance modelling process. The testing logic generated from the PSS based model controls all the operations between the UVC's. The IP level simulations are run using the generated tests and coverage is collected.

The Table 2 (below) shows the results of regression run with PSS and UVM based verification environment. The overall constrained random tests run to achieve the maximum coverage possible (with waivers) reduces significantly with the adoption of UVM based approach. The randomization mechanism in PSS based verification starts from an abstract description of the legal transitions between the high-level states of the DUT, and automatically enumerates the minimum set of tests needed to cover the paths through this state space. The graph-based coverage allows user to view the transverse paths and allows tests to be generated such that maximum length of the graph is covered.

The test quality is other factor which can be controlled better by the Portable Stimulus Verification Methodology. The test can be visually seen allowing users to understand the control and data flows in a better way. Also, some tools allow active checks to be put during the run-time allowing an effective automated checking. This would combine with the scoreboard checking and assertions-based check points improving the quality of verification.

The Portable Stimulus Methodology works at a higher layer of abstraction and then integrated with underlying verification process. Due to this, although there is a definite improvement in the test or stimulus generation process, this verification methodology would still inherit the underlying process in its original form. In case of integration with UVM based environments, on one hand it will be benefitted by the re-use of verification components, on the other hand, it will get limited by its complexity. Similarly, as is the case with UVM, the quality of verification is limited by the quality of the verification plan and analysis of the coverage reports.

Tests Run	Passed	Failed	Not Run	Overall Code Coverage
(UVM only) 125	125	0	0	298034/388949(76.6%)
(UVM PSS) 75	75	0	0	298034/388949(76.6%)

Table 2. PSS UVM Setup and Regression

C. SoC Verification of Interconnect Bus

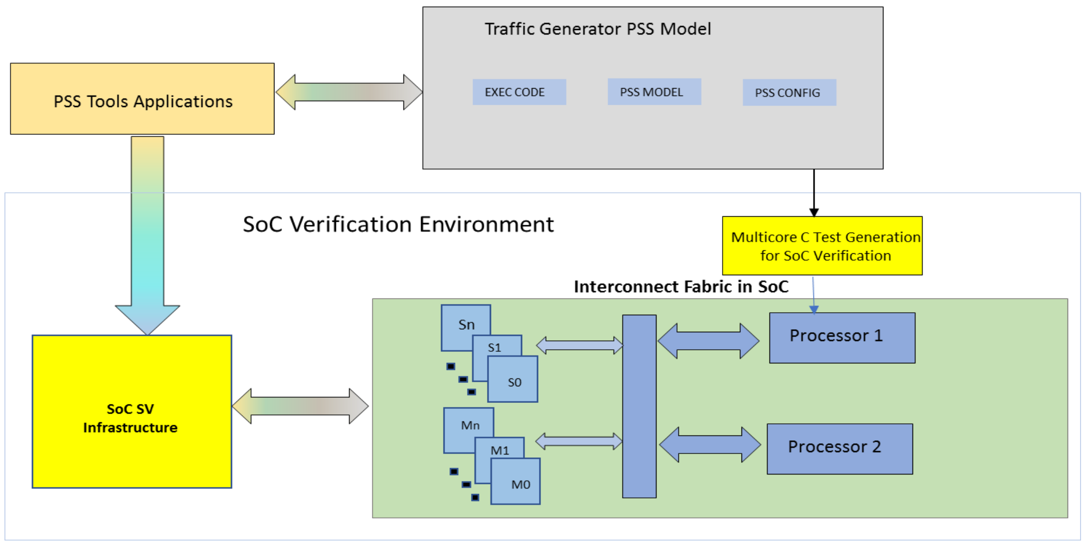

Traffic Generator PSS Model
EXEC CODE
PSS MODEL
PSS CONFIG
PSS Tools Applications
SoC Verification Environment
Multicore C Test Generation for SoC Verification
Interconnect Fabric in SoC
Sn
S1
S0
Mn
M1
M0
Processor 1
Processor 2
SoC SV Infrastructure

Figure 7: PSS based SoC Verification of Interconnect Bus

When the interconnect is integrated as a part of the SoC, it is vital to check its integration with the various masters and slaves in the system. This is typically done with the C-based tests that runs on the processor to check the integration of the Interconnect Bus. The generic masters at the IP level changes to specific Bus Masters like multiple processors, DSP, DMA Controllers, Serial protocol master like SPI, I2C, CAN etc. and many more custom masters and slaves. This asks for specific sequences or macros meant to control the different masters and the slaves in the SoC. The macros or sequences will typically have register programming to enable the transmit and receive transactions from masters like DMA Controllers, Memories etc. There is no constrained randomization at this level, so each scenario needs to be explored and written manually. In terms of reuse, some of the UVM Monitors from IP level can be used to monitor the protocol or Scoreboards to check specific points of interests. But, the test and the sequences which contains major part of the specification needs to be re-done in C with a different focus.

The PS based verification techniques are on the other hand designed for IP to SoC test reuse. The Fig. 7(above) shows the reuse of the Traffic Generator PSS models at the SoC level verification. The models coded at the IP level are configured for different address maps as per SoC specification and targeted for C test generation. The same set of sequences written at the IP level with the graph based constrained randomization are possible to reuse. Almost all the sequences in the model with exception of the parts meant for “Exec” code are reusable when we write model for the Processor based applications. Having said that, the “Exec” code in this case is substantially more including complete enable and disable macros for variety of masters like DMA and Memory Controllers which can initiate the single or burst transactions on the Interconnect Bus. For every generic master, the “Exec” code needs to be re-written so that it can integrate with the variety of masters in the SoC. The tool randomization allows multiple combinations of master and slaves transaction creating interesting scenarios. The constrains to create directed tests for integration checks at the SoC level can be managed well with the visual representation of the test. Once the C test are generated, they are integrated with the SoC setup with some system specific standard infrastructure. The C tests are then compiled and run on the processor to generate transactions.

The Fig. 7(above) also shows creation of multi-core tests with the PSS tools which are difficult to write manually. Different parts of test intent can be targeted to run on different cores allowing interesting scenario creation. It is specifically helpful in this case, where multiple bus masters exist including multiple processors. The programming of different masters becomes possible due to this feature set. The ability to reproduce graph based constrained random tests at the SoC level without the need to actual recoding of the scenarios is a major advantage as well. It also allows the test generation to different instances of the same IP with different address maps. Along with this, when different kind of PS models for different IP combined at the SoC level, the creation of complex scenario is possible which are otherwise difficult to code manually.

D. Validation of Interconnect Bus

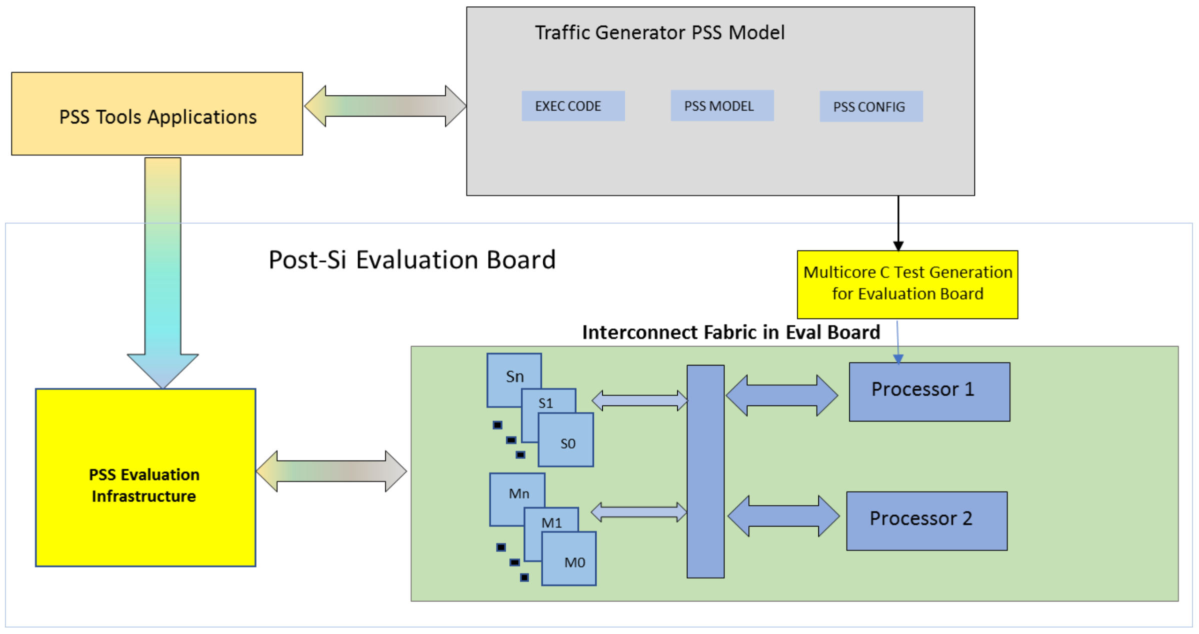

Traffic Generator PSS Model
EXEC CODE
PSS MODEL
PSS CONFIG
PSS Tools Applications
Post-Si Evaluation Board
Multicore C Test Generation for Evaluation Board
Interconnect Fabric in Eval Board
Sn
S1
S0
Mn
M1
M0
Processor 1
Processor 2
PSS Evaluation Infrastructure

Figure 8: PSS based Silicon Validation of Interconnect Bus

The validation process is required to establish product compliance to the customer specification, usability and acceptance testing. Traditionally, Evaluation board requires C based tests which are manually written from the original specification. This duplication of effort can be greatly reduced by using PSS based approach where C tests compatible with the evaluation software can be generated.

The Fig 8 (above) represents the Validation process with the PSS based approach. The PSS model for Traffic Generator can be configured for different address maps as per SoC specification and targeted for Eval-C tests generation. The PSS tools typically have ability to generate tests for multiple cores which allow specific scenario testing as well. The generated C tests are compiled by the debuggers and code is loaded on the Evaluation Board with interfaces like JTAG. The tests can be run, and results can be seen on the evaluation board and debugger interface. The same set of sequences written at the SoC level with the graph based constrained randomization are completely possible to reuse. In addition to this, the visual representation of the test intent and ability to apply constraints comes in handy to create directed scenarios. This is a unique and controlled way to create tests in the validation process which has traditionally been completely manual.

Here again, the "Exec" code needs to be re-written for Silicon Validation specific requirements. The basic software drivers from validation platform used to control different masters on the bus like DMA and Memory Controllers can be typically used for this kind of application. The generated C code also needs to be integrated in a format which is acceptable to the evaluation platforms. Typically, this process includes reusing the header and include files from a pre-written validation code and reusing it in the generated integration code. It is then compiled and run with the target debugger to ensure proper testing at this level.

The PSS tools typically provide the ability to analyze the results of the run with post processing applications. The visual analysis of the results indicating test pass or fail with specific segments of the code yielding results is possible. This is especially very helpful in validation process because traditionally debugging capabilities are very limited here.

Although, we haven't reused the C tests for Traffic Generator model at the Post-Si based application yet, but we are confident that it can be used on any evaluation platform employing C-based tests. Infact, this kind of models where SoC based PS models are reused for Post Si evaluation boards have been proven for other processor-based applications. This reuse is the kind of application which is unique to Portable Stimulus based approach only.

IV. SUMMARY

The PSS based Generic Traffic Generator allows test reuse for the Interconnect Bus from System-C based Performance Analysis to Verification and Validation process. There is a need for integration and infrastructure development for each of the process. But, this is a one-time process and offers a reuse possibility in the subsequent applications. Along with the reuse, the PSS based approach offers advantage in terms of specific randomization, visual representation of test intent and an early coverage analysis adding further value. The ability to create generic applications allows possibility of plug and play solutions which can further accelerate the verification and validation process.

REFERENCES

[1] http://www.accellera.org/activities/working-groups/portable-stimulus
[2] http://accellera.org/downloads/standards/uvm
[3] http://www.brekersystems.com/products/trekuvm
[4] http://www.synopsys.com/news/pubs/snug/2015/boston/F1.2_Ajamian_paper.pdf
[5] http://events.dvcon.org/2018/proceedings/papers/02_1.pdf