2014 DESIGN AND VERIFICATION DVCON CONFERENCE AND EXHIBITION EUROPE

A methodology for vertical Reuse of functional verification from subsystem to SoC level with seamless SoC emulation

Pranav Kumar, Staff Engineer (*pranav.kumar@st.com*)
Digvijaya Pratap SINGH, Sr. Staff Engineer (digvijay.singh@st.com)

STMicroelectronics, Greater NOIDA, INDIA

Ankur Jain, Verification Technologist (Ankur_Jain@mentor.com)

Mentor Graphics, Bangalore, INDIA

***Abstract*— Verifying a complex SoC is challenging. The testbench and testcases must be developed early as these are used for everything from SoC verification to achieve higher coverage on features/protocol coverage, i/f integration and performance verification. Time to market makes early software development a necessity. Verification assisted with early software development needs more than dynamic simulation and emulation methodology comes to rescue. Firstly, the verification of RTL in subsystem environment has been efficient with a SV/UVM methodology which focus on Reuse on testbench and testcases along with the verification environment around Verification IP. The Reuse methodology in center, the emulation methodology offers a huge theoretical performance advantage over dynamic simulations. Theoretically, too often verification teams get bogged down spending enormous amounts of time porting verification infrastructure from one level of testing to the next. It is prone to testbench bugs also. Beyond the horizontal and vertical re-use model, there is a need of end to end re-use across the design and verification flow. This paper explains a generic methodology to achieve this end-to-end reuse in an ongoing project.**

Keywords—functional verification, emulation, SV/UVM, Lightweight virtual platform, LVP, dynamic simulation, subsystem verification, SoC verification, early software development, system level verification

I. INTRODUCTION

The current methodology focuses on functional verification of high speed interface subsystems and these are USB3/2, PCIe, SATA and GMAC Ethernet. The PCIe subsystem has been deployed in this context, in the process developing a mature verification methodology for project cycle i.e. SoC design, verification and implementation. The need for simulation speedup and various ways to achieve the same has been discussed [2]. The speedup of simulations is must noticing the SoC increased complexity and another way to achieve simulation speedup is scaling down with virtual platform which has a processor model and facilitates a software driven verification environment [2][3]. The verification environment has the verification components in SV/UVM from a Verification IP library and these components fit nearly any verification environment. These verification IPs delivers a common interface across the library of protocols which has been very helpful going from one protocol to other protocol. This results in a scalable verification solution for high speed interface protocols and standard interfaces; the solution includes stimulus generation, reference checking, and coverage measurements that can be used for RTL, TLM, and system-level verification. The SV/UVM environment has a SV testbench and verification IP is integrated along with a wrapper giving a software interface to verification environment [1]. In addition, emulation ready verification IP with similar interfaces help achieve the emulation setup for subsystem or SoC stress scenarios.

As depicted in Figure 1, the verification starts at subsystem level. Vertical Reuse while SoC verification is facilitated by what is produced for subsystem verification. The focus is on functional verification environment which is based on a Lightweight Virtual Platform (LVP) which is a scale down version of a SoC at a high level of abstraction as described. Emulation ready verification IP helps Horizontal Reuse and intent is to reuse yet again. Efforts and time at all these levels of testing can be reduced if testcases and testbench can be reused across dramatically at various levels e.g. Block/Subsystem and SoC simulation, emulation and during validation i.e. hardware/software testing.

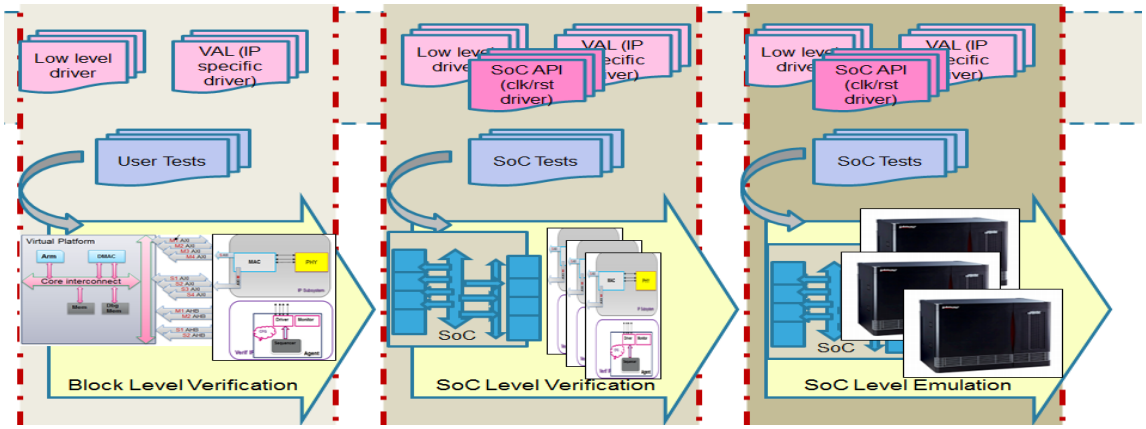


Figure 1 : Vertical Verification to Extend Horizontal Reuse

The subsystem team designs and verify high speed interfaces e.g. PCIe, and then delivers their designs, along with reusable verification infrastructure to teams performing SoC verification, emulation and early-stage software testing. Real examples of testbench architecture and code snippets will be used to illustrate the implementation.

II. SUBSYSTEM LEVEL VERIFICATION ENVIRONMENT

A. *Scaling down the problem*

The rise of SoCs has led to manifold increases in complexity. In first place, verifying complex SoC sub-components has become difficult due to limitations in simulation speed and debug capability. Hence the effort to scale down the problem, as depicted in Figure 2. The simplified lightweight virtual platform (LVP) with AMBA transactors is the key component in the verification environment. An ARM core ISS wrapped in SystemC/TLM permits us efficient debugging of software (including tests, low level drivers, etc.). The memory hub is modeled at a higher level of abstraction in SystemC/TLM and the I/O hub is the partition for the RTL subsystem. The bridge/transactors are used to connect the memory and I/O hubs.

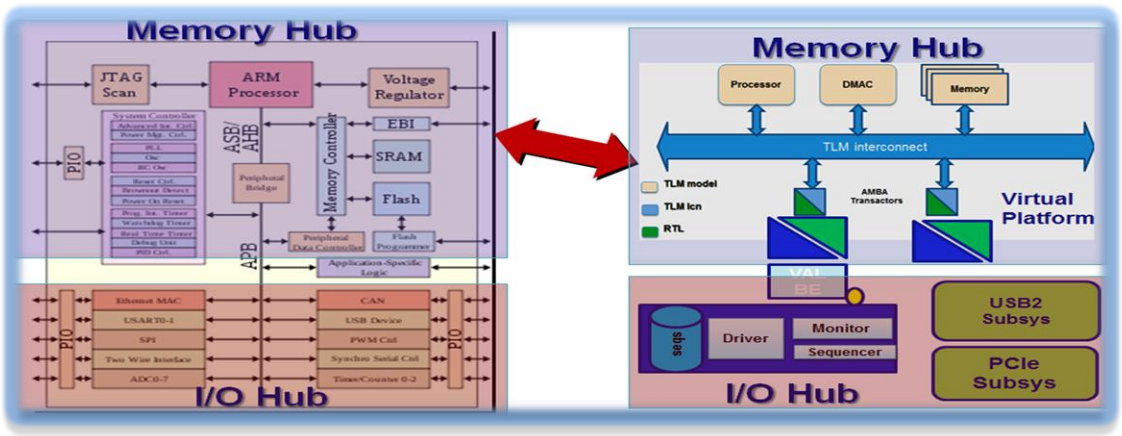


Figure 2 : Scaling down the problem

B. *Verification environment*

Benefits of the scaled down environment has ease of use and benefits w.r.t. simulation performance and also for the ease of use as the initialization code or boot code for this kind of environment are minimal. The verification environment depicted in Figure 3 has been deployed. The LVP enhanced with TLM transactors which permits to make transition from tlm to signal level are deployed for various AMBA protocols e.g. AXI, AHB, APB and wherever required for STBuses (T1, T3)[*]. The memory models in the environment are abstracted TLM models with backdoor access capability. This feature has been deployed heavily for

[*] STMicroelectronics proprietary.

scoreboarding in the environment. The C tests developed has been layered appropriately as depicted in Figure 4. The basic idea of layering is easy integration of the subsystem tests into SoC verification. In SoC context, there are certain configurations for clock/reset and interrupt handling which shall be taken care patching some function calls in the tests. Another important change is w.r.t. memory map as moving from subsystem to SoC context, memory mappings are changed or bound to change. The layered architecture of software ensures instrumentation of some header files and just make the memory changes to these locations as per SoC context.

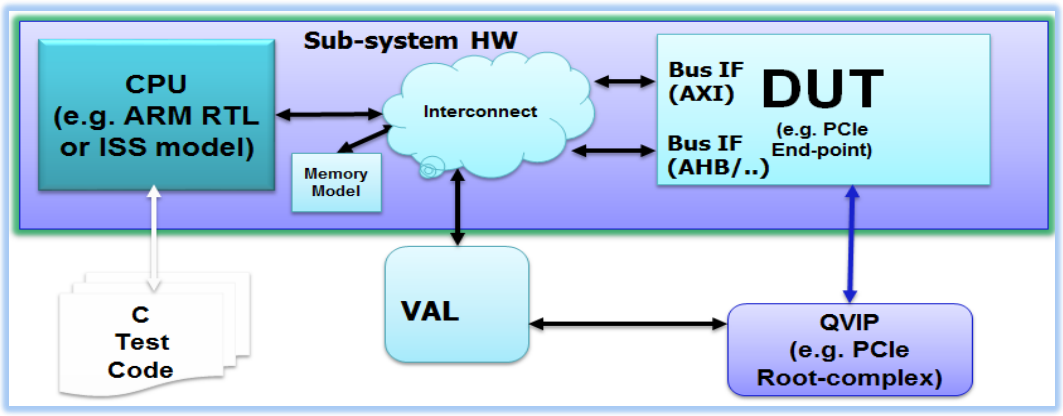


Figure 3 : Verification Environment @Subsystem level

Such a verification environment also enable the early development of low level driver APIs and SW test code at the block level and testing of PCIe low-level driver progress here onwards. The hardware IP (RTL) is packaged with software IP which include low level drivers, the sequences/uvm tests and other reports of subsystem level verification. The Re-Use at higher levels of integration is based on the packaged database.

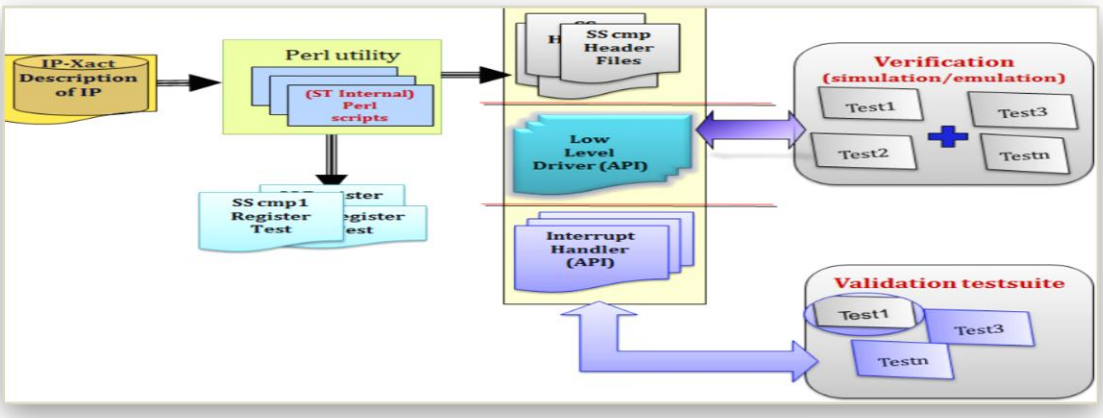


Figure 4 : The software architecture facilitates subsystem to SoC verification

The verification environment is based on virtual platform which is developed with TLM/SystemC component, SV/UVM verification IPs with a similar interface across various protocols. There are some other standard tools and platforms deployed e.g. GCC, SV/UVM and FastModel license. The SV/UVM Verification IPs are straight forward, needs to be simply instantiate as a component in the testbench. Mentor Verification IP delivers a common interface across the library of protocols. This results in a scalable verification solution for popular protocols and standard interfaces, including stimulus generation, reference checking, and coverage measurements that can be used for RTL, TLM, and system-level verification. Verification with Mentor VIP is straight forward: simply instantiate it as a component in your testbench, along with required environment configuration.

C. *Interface to verification environment*

In the case of subsystem verification, the software runs on a processor and the DUT is available to configure/control through AMBA bus transactors. However the software doesn't have control of the

verification environment as the verification component is external to the SoC. A verification component or VIP for high speed subsystems (PCIe) typically connects to the DUT on the serial interface. The limitation is that there is no software control of verification. In the subsystem environment, a wrapper around the verification component is created that not only wraps the verification component but also provides a way to generate events for controlling verification environment overall.

To leverage the availability of the AMBA bus transactor, a wrapper is interface created around the verification component so software can interact with the verification environment. Using this interface, software can configure verification IP and otherwise the control verification environment (generate events to perform a reset or stop VIP, or enable/disable scoreboarding). Hence, the wrapper layer, here called the Verification Abstraction Layer (VAL), enables complete control of the verification environment and not just configuration help. In addition, the software capability of event generation allows for software-driven constrained random configuration of VIP at runtime.

The limitation or unavailability of the AMBA interface in the SoC context is acknowledged in the VAL architecture which permits replacing the front end with any available interface including the SRAM or external memory interface. The VAL layer is depicted in Figure 5.

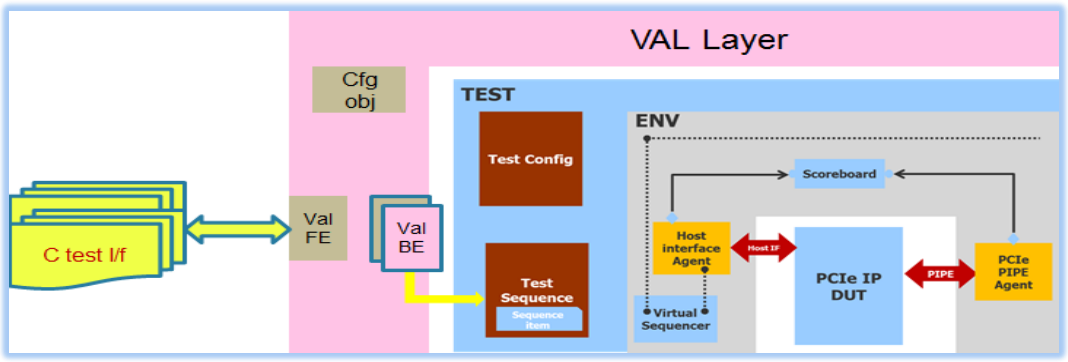
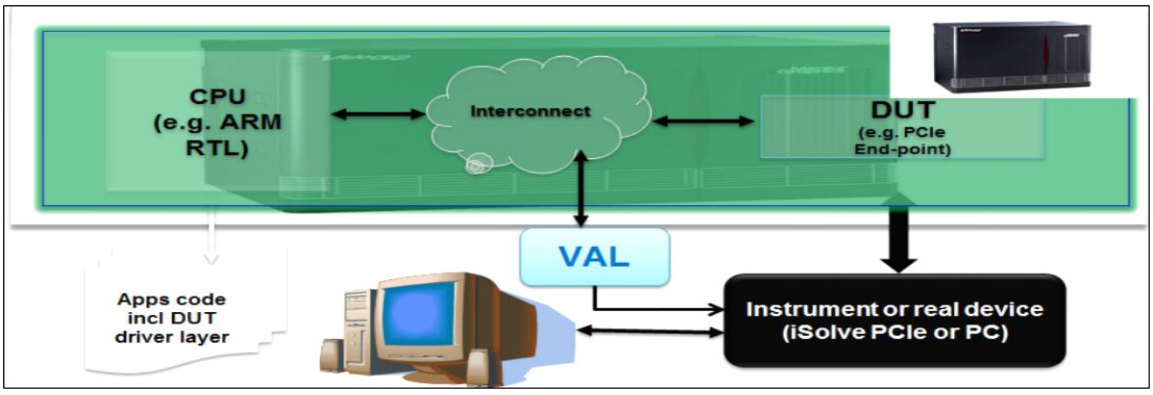


Figure 5 : Verification environment (incl. VIP) wrapper to interface with S/W

D. *Emulation Environment*

One reason to deploy the SV/UVM verification IP is to have emulation ready verification IP. Hence, the VAL layer which is the bridge between the software and verification environment need to be the key component ensuring seamless porting of verification environment.



The approach described in [8] has been deployed in the subsystem verification and emulation context. The PCIe subsystem RTL has been ported in emulation environment alongwith all synthesizable design components. Emulation is a technique that allows the synthesizable portion of the design to run on an emulator and the rest of the verification environment running on a legacy simulator.

III. TESTBENCH ARCHITECURE

The testbench in subsystem environment mainly consists of following components –

- **LVP** is a processer subsystem (SystemC/TLM) with multiple AMBA interfaces bridge to control the DUT & VIP. The SystemC model is wrapped into a Verilog shell and instantiated in the systemverilog testbench. The LVP consists of abstracted interconnect which is equivalent to ST's own library equivalent to TLM 2.0. The abstracted C++ memory models are integrated in the SystemC/TLM and DMAC, VIC tlm models are integrated in the virtual platform. The DMAC in the platform permits to extend the verification scenarios where SoC DMAC is performing bulk data transfers with the subsystem (PCIe subsystem).
- **DUT** is PCIe subsystem (the integrated entity consisting of PCIe controller core, PHY, memories and wrapper blocks along with clock and reset control).
- **VAL** (Verification Abstraction Layer) provides control, triggering and synchronization of UVM VIP -configuration, randomization and sequences from a C testcase running on the LVP. Also, VAL is in-charge for providing control of other testbench components e.g. scoreboard.
- **VIP** is connected to LVP through VAL via AMBA interface where AMBA i/f can be replaced with any other bus i/f. In the SoC environment, the interface is replaced with memory i/f.

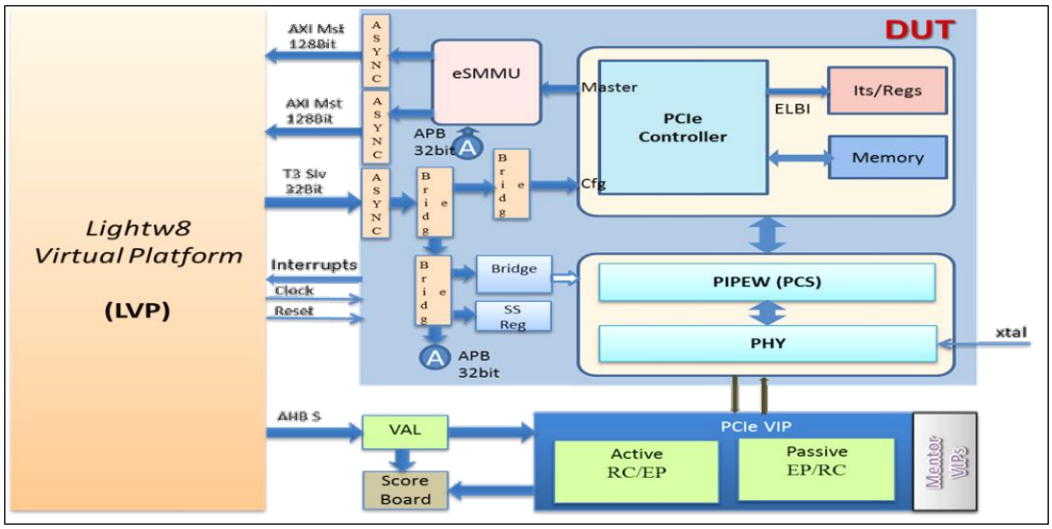


Figure 6 : Testbench in the verification environment

A. *VIP as an IP*

One of the important requirements for the vertical and horizontal re-use of the verification environment is the portability of the VIP along with its configuration and control from subsystem to SoC level or for emulation setup. To achieve this easy portability the verification component is considered consisting of custom defined registers set with VIP driver development in 'C', just as done for any other IP or subsystem.

It helps to visualize the verification component on the same line as for any IP or subsystem integrated in SoC. The major configuration parameters of the VIP are identified and C data-structure defined for the same. All functions for the major VIP activities viz-a-viz configuration/control and read/write transactions, etc., are developed.

```
typedef union {
  struct {
    uint32_t tlp_type : 8; // [7:0]
    uint32_t reserved : 7; // [14:8]
    uint32_t bar_size : 1; // [15]
    uint32_t unused_0 : 16; // [31:16]
  } BitI;
  uint32_t RegI;
} t_PCIe_QVIP_TLP_ID_REG;

typedef union {
  struct {
    uint32_t seq_id : 32; // [31:0]
  } BitI;
  uint32_t RegI;
} t_PCIe_QVIP_SEQ_ID_REG;
……
....

typedef struct {
#ifdef BLOCK_LEVEL
  uint32_t* VAL_TRIG;  // @0
#else
  uint32_t VAL_TRIG;   // @0
#endif

  t_PCIe_QVIP_SEQ_ID_REG          PCIe_QVIP_SEQ_ID_REG;       // @4
  t_PCIe_QVIP_SEQ_ADDRL_REG       PCIe_QVIP_SEQ_ADDRL_REG;    // @8
  t_PCIe_QVIP_SEQ_ADDRH_REG       PCIe_QVIP_SEQ_ADDRH_REG;    // @12
  t_PCIe_QVIP_SEQ_DATA_LEN_REG    PCIe_QVIP_SEQ_DATA_LEN_REG; // @16
  t_PCIe_QVIP_SEQ_DATA_REF_REG    PCIe_QVIP_SEQ_DATA_REF_REG; // @20
  t_PCIe_QVIP_PLP_REG             PCIe_QVIP_PLP_REG;          // @24
  t_PCIe_QVIP_END_SEQ_REG         PCIe_QVIP_END_SEQ_REG;      // @28
  t_PCIe_QVIP_SYNC_FLAG           PCIe_QVIP_SYNC_FLAG;        // @32
  t_PCIe_QVIP_MSI_OFF             PCIe_QVIP_MSI_OFF;          // @36
  t_PCIe_QVIP_DLLP_REG            PCIe_QVIP_DLLP_REG;         // @40
} PCIe_QVIP_registers;
```

The Verification component configuration has been arranged in register set and made word addressable 'C' declaration for registers

The register set for Verification IP are mapped to consecutive addresses in memory, reserving 0^{th} for VAL triggers.

Figure 7 : VIP registers description in C

B. *C & SV/UVM Synchronization*

The key challenge here is the interfacing of the VIP driver which is in 'C' to the VIP environment which is in SV/UVM. This interface is described as VAL or Verification Abstraction Layer. The same VAL is used for interaction with scoreboard (UVM world) for self-checking capability.

The VIP configuration is based on SV/UVM config classes and the configuration takes place on appropriate trigger through VAL. The sequence selection to be run is also controlled from C-test.

An important aspect to be taken care is that the VIP needs not to start before proper configuration of the IP and VIP itself. The required synchronization is achieved using tlm_block port, as the VIP sequencer is blocked till trigger doesn't arrive from C-test via VAL. This implementation is demonstrated in figure 7 below.

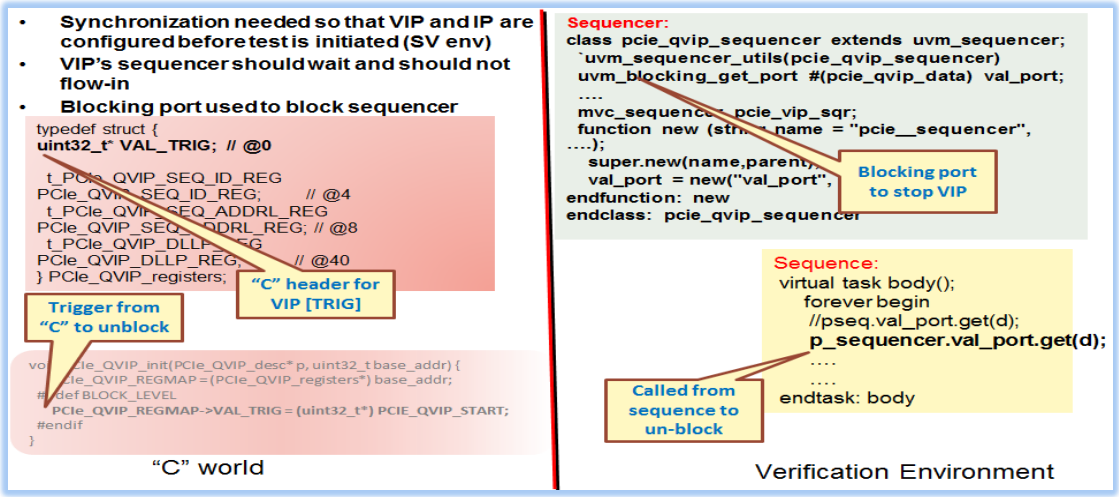


Figure 8 : C and SV/UVM synchronization example.

C. *Configuration and Data Flow Control via VAL*

As described earlier in the section, VAL is developed as a generic configurable component with two sub-layers:

a) The VAL_FE which is taking of the interface to any standard (AMBA, STBus, etc.) or proprietary bus or TLM model.

b) The VAL_BE which is converting the instructions received from VAL_FE to various configuration or data packets and sequence generation or control of existing sequences.

The Figure 8 shown below describes two VAL_BE interfaces – QVIP_VAL_BE and TEST_VAL_BE. The QVIP_VAL_BE is interfaced with Questa-VIP and TEST_VAL_BE is interfaced with Coverage-collector and Scoreboard.

- The QVIP_VAL_BE has two ports – config and data. The config_port is responsible for the configuration/control packets of the verification component. The configuration can be randomized and applied to the verification component as and when needed. The data_port is controlling data transactions along with required control of sequence generation and the running of sequences on the sequencer.
- The TEST_VAL_BE has two ports – test_config and test_data. Test_config is responsible for randomization of the test/config parameters for the required test and their transfer to main memory and coverage-collector. These randomized test parameters are read by the c-test from memory for the VIP and IP configuration.

  The test_data port is responsible for providing reference and test data to scoreboard for the self-checking capability.

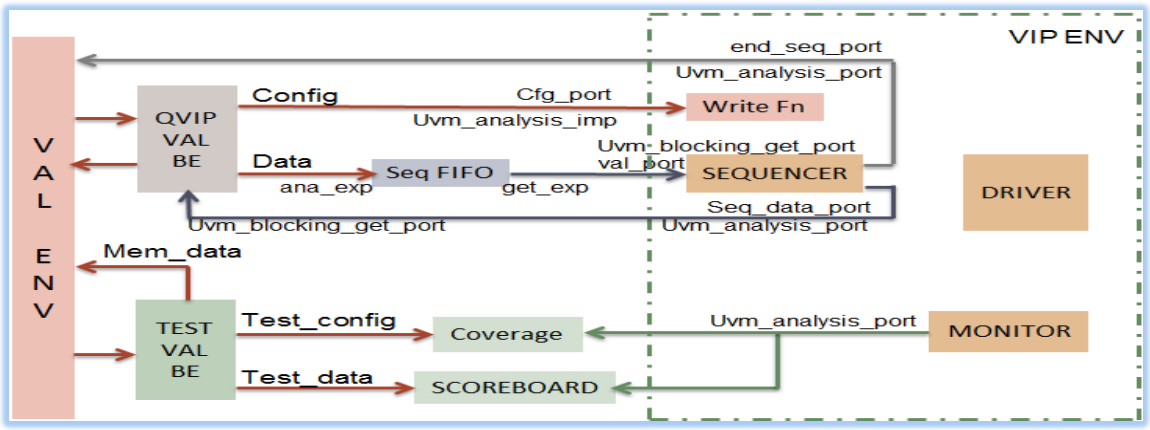


Figure 9 : The Config and Data management in verification environment

IV. CONCLUSION

The proposed methodology has been very helpful in SoC projects with high speed subsystems. Early development of the testbench and testcases on the basis of the testplan was efficiently achieved and these components were made available by the time SoC integration has taken place. Also, the library of early software drivers and tests were packaged and deployed for further use on SoC subsystems. The integration tests were identified that are likely to be useful in the context of SoC verification and hence suggested to the SoC team. It helps to focus only on useful tests to check SoC integration; rest is taken care of in subsystem level verification.

Prototyping results from emulation have been smooth and encouraging except very few surprises during emulation implementations. From SoC verification to emulation, very few changes in the environment are required; hence, verification effort is minimized.

Another value addition to project is reuse of software drivers allows for earlier validation of silicon and software. The set of tests developed for the subsystem level are used to validate PCIe in silicon with the same software drivers developed during validation.

V. FUTURE WORK

The methodology has proved to be helpful with very rapid migration to SoC verification from subsystem level. The re-use of subsystem verification environment with Verification Abstraction Layer (VAL) leveraged without any major changes or development needs in SoC context. As a subsystem team, the deliverables are packaged with SV/UVM sequences, testbench, uvm tests and "C" test-suite.

The emulation activity has been evolving with the deliverables done at subsystem verification level. In future, the tests will be completely integrated in emulation environment and regressed. Another important task would be to re-use the subsystem drivers and tests to create application scenarios involving simultaneous working of various IPs in SoC. Such a kind of verification is not feasible in simulation environment and emulation can prove phenomenally good for all such scenarios.

Another important objective is that software drivers developed during functional verification at subsystem level also get re-used for silicon validation. The experiments with earlier SoCs with similar IP prototypes has proved to get good results, hence shall be considered for next generation of SoCs.

VI. ACKNOWLEDGMENT

We would like to thank Alberto Evasio Allara for his technical support and guidance. Further, we extend thanks to Rajiv Kumar for all support on project needs. We would like to convey special thanks to Nitin Sharma and Vinit Shenoy on methodology support. We would like also to thank our colleagues of STMicroelectronics and Mentor Graphics.

REFERENCES

[1] Alberto Allara, Fabio Brognara "Bringing constrained random into SoC SW-driven verification" DVCon SanJose, 2013

[2] Gordan Allan; "Tried and tested speedups for SW-driven SoC simulation", DVCon, 2014.

[3] Matthew Balance, "Boost Verification Results by Bridging the Hardware/Software Testbench Gap" DVCon 2013

[4] HU Zhaohui, Arnaud PIERRES,HU Shiqing, Chen Fang, Philippe ROYANNEZ, Eng Pek SEE,Yean Ling HOON, "Practical and Efficient SOC Verification Flow by Reusing IP Testcase and Testbench" , SoC Design Conference (ISOCC), 2012 International

[5] Hunter, A. ; ARM Ltd., Austin, TX ; Piziali, Andrew ; Ziv, A. ; Larson, K., "Ensuring Functional Closure of a Multi-core SoC through Verification Planning, Implementation and Execution", Microprocessor Test and Verification, 2008. MTV '08. Ninth International Workshop on

[6] Klein, R. ; Mentor Graphics Corp., Wilsonville, OR, USA ; Piekarz, T., "Accelerating functional simulation for processor based designs", System-on-Chip for Real-Time Applications, 2005. Proceedings. Fifth International Workshop

[7] Digvijay SINGH, Sandeep Jana, "A generic, cohesive approach for IP and SoC verification using Virtual Platform" TechOnline – August 30, 2012

[8] Alberto Allara, Lanfranco Salinari et al; "Simulation + Emulation = Verification Success", Mentor Graphics and STMicroelectronics.

[9] www.systemc.org

[10] www.arm.com

[11] https://verificationacademy.com/