Unified Approach for Performance Evaluation and Debug of System on Chip at Early Design Phase

Nishit Gupta, Sunil Alag
R&D in Electronics Group, Department of Electronics & Information Technology
Ministry of Communications and Information Technology, Government of India
New Delhi, India

***Abstract*—This paper proposes a novel approach for System Level Debug and Performance Evaluation that exploits the signal level and clock cycle accuracy existing in Bus Cycle Accurate hardware IP models along with the advantages of untimed Transaction Level Modeling. The developed toolset can be integrated in SoC simulations in a nonintrusive manner which secretly embeds performance figures and debug information in dumped simulation database at signal and transaction level. Proposed approach suggests modeling the SoC components with only functional accuracy in which the computational delays are added using the timing features provided by event based SystemC kernel. The components are modeled with clock cycle and signal level accuracy at the interface. Profiling results shows that the proposed approach outperforms several state-of-art methodologies in terms accuracy, adaptability and simulation speed by an order of magnitude of 10^2. The developed toolset can effectively be used in a co-simulation environment with IPs at different abstraction levels.**

Keywords— AXI; Bus Cycle Accurate; Latency; SystemC; System on Chip; Transactional Level Modelling

I. INTRODUCTION

Traditionally, RTL is the de-facto standard for optimizing the architecture of System on Chip to meet the desired performance. Only after the RTL of SoC is made available, the System Architects start extracting the performance figures from SoC simulations. The SoC RTL is also deployed for extracting Power figures and running embedded software for verification purpose. Considering the huge delay that occurs in writing the complex register-accurate, clock-cycle accurate and bus-cycle accurate RTL IP models and high simulation time, many efforts has been made [4, 5, 6, 8] in the last decade to raise the abstraction level of SoC modeling for making available a lighter version of SOC for System Architects capable to be exploited for performance evaluation. Transactional level modeling, arguably, is the best approach for running the embedded system-level software for functional verification but being approximately-timed with no bus-cycle accuracy, such models cannot be deployed for performance estimation of SoC. Bus Cycle Accurate IP models with functional part modeled at Transaction Level (TLM) with approximate timings and Bus communication part modeled at signal level can be effectively used for Performance evaluation. SystemC, which is a C++ class library, specifically designed for hardware modeling, can be used efficiently for developing such BCA IP models as it provides timing notion, hardware data-types and thread library. It also provides other C++ features like polymorphism and inheritance which can be exploited to developed re-configurable IP models.

In this paper, an approach to extract performance figure like latency, bandwidth, throughput, occupancy, opcode table from SoC simulation is proposed. The toolset developed, secretly converts the signals into transactions and adds debug information which can be used for finding ambiguous behavior in simulation. It also dumps the traffic characterization file which can be used to provide stimulus to initiator IPs in the absence of RTL. It allows a system designer to take system level design decisions in very early stages of system design and hence avoiding redesign efforts and performance bottlenecks in advanced stages of a project. Another advantage is the fast simulation speed of such models enabling running many use cases in a short span of time in comparison to RTL.

II. RELATED WORK

While the concept of deploying Bus Cycle Accurate IP models for performance estimation is not new, to the best of our knowledge, this is the first implementation of a complete system level mixed flow exploiting the advantages of Transactional Level methodology and Bus Cycle Accurate models both, in a co-simulation environment with IPs at multiple abstraction levels. The proposed approach not only provides a toolset to extract performance figure out of the TLM-BCA models but also provides necessary debug information at transactional level. There have been similar works in the literature which talk about software architecture platforms implementing a flow exploiting Bus Cycle Accurate abstraction level for performance estimation of a specific IP or sub-system [1, 4, 12] but they did not leverage the advantages of high simulation speed provided by TLM models deployed with accurate BCA models. Further, in the absence of any debug features available at signal level BCA flow currently used commercially or for research purpose, it's very hard to debug the complete system which raises a certain doubt on the results obtained from of such approaches.

III. SYSTEMC AS A MODELING LANGUAGE AT MULTIPLE ABSTRATIONS LEVEL

The functionality of any hardware IP can be broadly classified in computation phase and communication phase. While in computation phase, the IP could be reading/writing the data, waiting/generating an event/interrupt or doing no operation. The computation phase of IP can be modeled using event based synchronization, wait and notify calls provided by

978-1-4673-7948-9/15/$31.00 ©2015 IEEE

SystemC kernel[3, 11, 13]. To model the communication delay, IP is modeled with accuracy at signal boundary acting on clock trigger. As the computation phase is modeled using few SystemC calls rather than the complete combinational logic acting on the trigger of clock, the context switches taking place are very low in number and thus a high speed is obtained using this approach. An important step in performance simulations is modelling the IP behaviour for a given usecase. It is desired that IP modelling could be done as quickly as possible (for its different modes). In many cases IP's or its software drivers are not available at the time of interconnect freeze, thus using the actual IP for analysis is not possible. Following section briefly describes an efficient IP modelling example relying on data in IP data sheets like bandwidth, opcode used, latency etc. The modelling is done for an IP which generates a constant average b/w. After reset it starts reading data from the DDR interface at a frequency of 1000 MHz and reading 64 bytes on each clock which is the data bus size of AXI bus. After reading 5120 bytes the IP will come to stall. Traffic characterization file, as in Fig. 1, is provided as input to automatically generated BCA models of Bus Masters to produce realistic traffic on interconnect. Further, Smart Pointer in SCV Library, as in Fig. 2, provides a probability distribution curve as in Fig. 3

```
$IP_NAME                D_MAC
$IP_INTERFACE           AXI
$IP_FIFO_SIZE           2048
$IP_FREQ                1000MHz
##------------------- 1 ---------------------------------------
##  Description: wait for the system to initialize
##-------------------------------------------------------------
$PROCESS_NAME       init
$PROCESS_SEQ        NOP 500ns
##------------------- 2 ---------------------------------------
##  Description: read from random address locations
##-------------------------------------------------------------
$PROCESS_NAME       read_from_DDR
$PROCESS_OPC        READ
$PROCESS_ADDR    {0x0~0x50}=40;{0x60~0x90}=60
$PROCESS_DATA_LENGTH          512*10
$PROCESS_PEAK_BANDWIDTH   100MBps
```

Fig. 1. Input configuration file for trafic characterization

```
scv_bag<pair<int,int> > addr;
addr.add(pair<int,int>(0x10,0x50), 20);
addr.add(pair<int,int>(0x50,0x90), 80);
scv_smart_ptr<int> ptr;
ptr->set_mode(addr);
ptr->next();
```

Fig. 2. SystemC Verification Library (SCV): Smart pointer for randomization

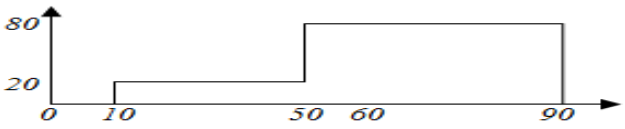

Fig. 3. Probability Distribution obatined from SCV Smart pointer

```
sc_trace_file *ptr;
ptr=sc_create_vcd_trace_file("wave");
sc_trace(clock,clk,"clk");
sc_trace(input,din,"din");
sc_trace(output,dout,"dout");
sc_close_vcd_trace_file(ptr);
```

Fig. 4. Tracing internal signals with SystemC

```
.clk("sc_main.TOP.MES_RTL_SHELL.clk_stbus") \
.req("sc_main.TOP.MES_RTL_SHELL.targ_req")  \
.gnt("sc_main.TOP.MES_RTL_SHELL.targ_gnt")  \
      .....
.id("sc_main.TOP.MES_RTL_SHELL.targ_id ")     \
.record(on) .bus_size(64 .perf(on) .debug(on)  |
.prt_check(on)
```

Fig. 5. SyPerf connections with internal signals in Sysperf.cfg

SystemC provides a mechanism [9] to dump internal variables and interface signals and port values in VCD format as shown in Fig. 4. The performance figures and debug information is computed at the simulation time and is dumped in the simulation database at signal level or transaction level which can be viewed in graphical viewer like Simvision.

IV. PLATFORM GENERATION

The platform can be generated and assembled automatically using the in-house developed GUI SoCKit. A typical TV SoC assembled through SoCKit looks as shown in Fig. 6. The analyzer tool "SysPerf" developed in-house is used to extract performance figures and is enabled through a configuration file sysperf.cfg as shown in Fig. 5 which expects the complete hierarchy of SystemC signals to be specified in cfg file where the probing has to be performed.

V. SIMULATION AND ANALYSIS METHOD

Performance simulation platform namely SAP, as shown in Fig. 7, consists of the following modules:

a) SOC interconnect BCA model in SystemC

b) Memory controller and memory models (in HDL) instantiated at the target ports of the interconnect model wrapped in a top level systemC file (to enable co-simulation)

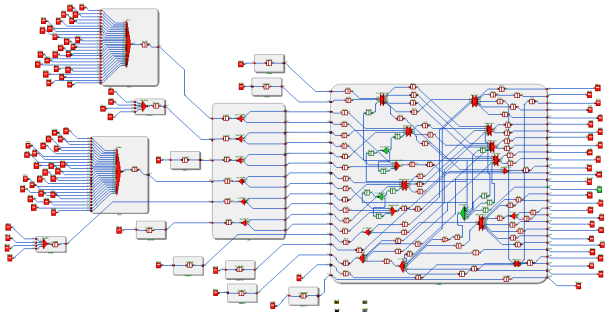

Fig. 6. A typical interconnect

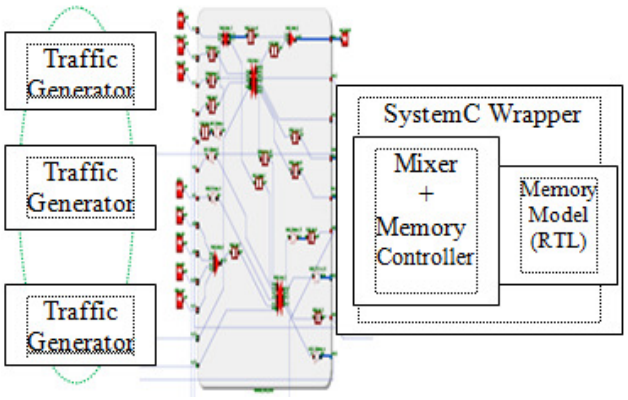

Fig. 7. Software Architecture Platform (SAP)

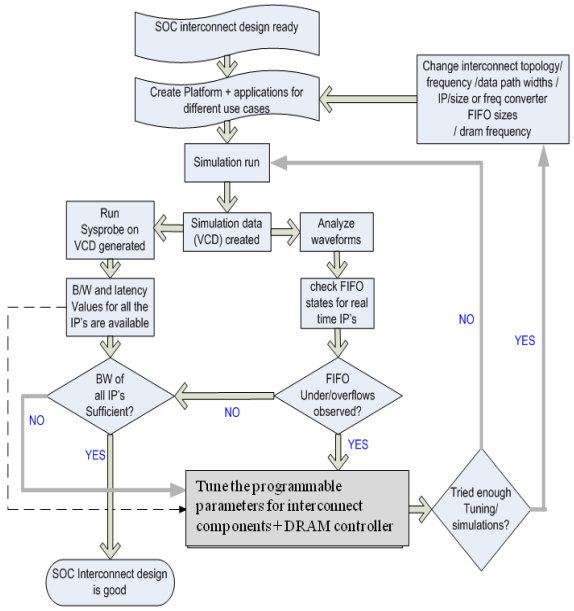

Fig. 8. Performance Simulation Flow

c) Generic traffic generator model (in System C) instantiated at the initiator ports of interconnect top level

d) Collection of traffic generator configurations (called "application" for a given use case) mapped to the corresponding system C file at the top level.

Once the platform and applications are created, the next step is simulation and analysis. A standard simulation tool (cadence ncsim) is used for simulation and SysPerf is used for analysis. SysPerf is enabled by following steps:

a) Set the name of output database

setenv SYSPERF_TX_DB waves

b) Set the probing file

setenv SYSPERF_CONFIG sysperf_config.cfg

c) Set the installed library path

setenv SYSPERF_LIB_PATH <installation path>

d) Dynamic analysis flow is added to an existing simulation non-intrusively. The elaboration command of cadence IUS simulation should be modified to load the Sysperf library.

ncelab –loadsc top_ncsc.so sc_main <options>

↓

ncelab –loadsc top_ncsc.so -loadsc ${SYSPERF_LIB_PATH}/lib/libsysperf.so sc_main dynamic_perf_top <options>

e) The simulation is run by command below:

ncsim sc_main <various_options>

Fig. 8 shows the steps involved in performance simulations and analysis. Interconnect design, simulation platform and traffic generator models and applications are created after the SOC architecture specification is ready. Platform simulation is then carried out which gives the waveform VCD dumps for subsequent analysis. If the traffic generator models and applications are ok, this step simulates the system traffic for a given use case. There are two kinds of simulation dumps created, i.e. "fifo_analysis.vcd" and "simulation.vcd". "fifo_analysis.vcd" gives the initiator internal FIFO levels. As shown (Fig.9, arrow 1) the IP starts filling the internal FIFO very late due to which it remained blocked as no request can be generated on interconnect and after that, it produces data at a very high rate. Due to the high latency of the system, the FIFO gone overflow (Fig. 9, arrow 2) as interconnect is not accepting requests. Simulation results clearly show that the FIFO level and rate of production of data inside IP needs to be re-programmed. SysPerf tool analysis on simulation.vcd gives vital information like bandwidth and latencies for a given IP (and at different levels inside interconnect) which are very useful for identifying the issues in the interconnect design and also to find the right arbitration parameters and bottlenecks. As an output, SysPerf also provides performance log as well as transactional dump, with debugging information embedded in it, out of the interface signals. SysPerf can also be executed on the simulation dump produced post-simulation to extract performance log and transactional database for debugging.

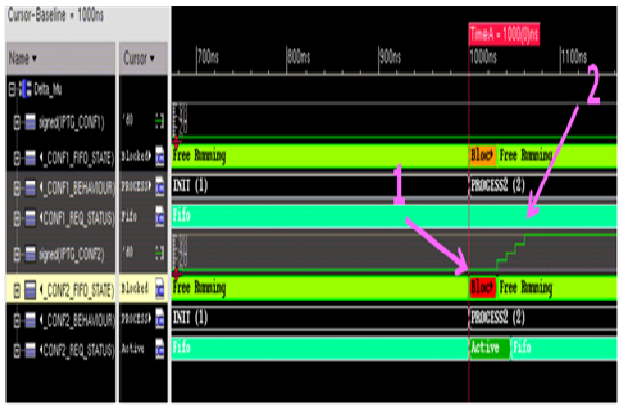

Fig. 9. Simvision display of fifo_analysis.vcd

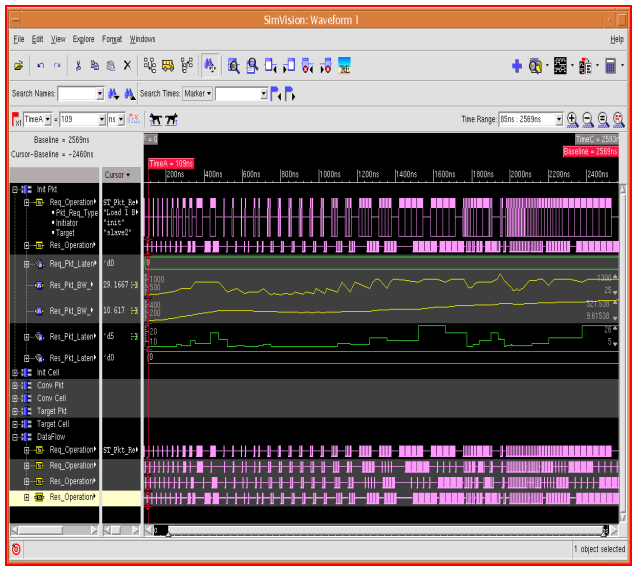

Fig. 10. performance figures produced by SysPerf

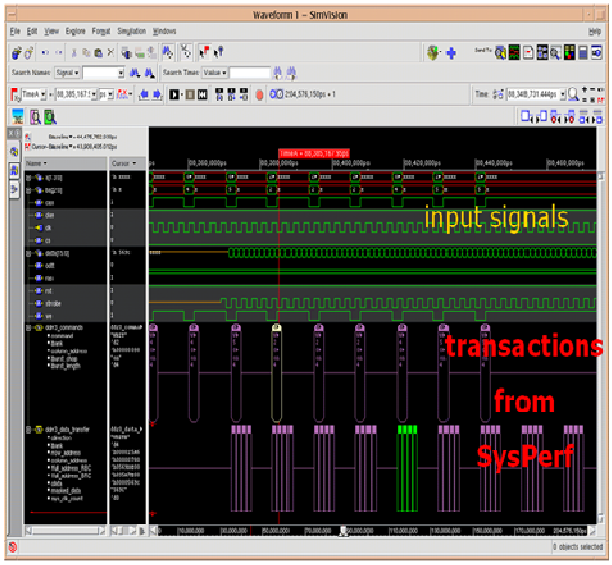

Fig. 12. Transactional database produced out of signals for debugging

The probing tool extracts performance log as shown in Fig. 11 from the simulation.vcd file. The average Bandwidth obtained from the first simulation run of SoC BCA models when compared with actual SoC RTL simulation shows a relative error of 6.74%. This reflects a very high accuracy as compared to spreadsheet analysis or high level C++ simulation results. It is essential to correlate the predicted performance with real performance figures measured on the silicon device refine the models later. SysPerf tool also provides rough estimates of power figures in text format and VCD file

PERFORMANCE LOG

Simulation duration: 189700000 fs
Clock period: 10000000 fs
AXI3 Frequency: 100 Mhz

LATENCY

Name	*Min*	*Max*	*Avg*
awvalid to awready	*0*	*0*	*0*
arvalid to arready	*45*	*45*	*45*
rvalid to reready	*5*	*5*	*5*

AXI3 PIPELINE

Name	*Min*	*Max*	*Avg*
Burst Pipeline	*2*	*2*	*2*
Beat Pipeline	*4*	*4*	*4*

AXI3 OPCODE TABLE

Opcode_table with byte Enable
*Load 1 Byte (1 bytes *3) = 3 bytes(0.68%)*
*Store 1 Byte (1 bytes *2) = 2 bytes(0.45%)*
*Load 2 Bytes(2 bytes *11) = 22 bytes(5%)*
……..
Number of Bytes Stored 443

BANDWIDTH

Max. Available Bandwidth 40MB/s
Min. Available Bandwidth 8 MB/s

Fig. 11. Performance log

associated with communication phase of IP as shown in Table I. The total power consumed at any instance of time is the sum of the power consumed by both the threads as shown in Fig. 13. From the results it was educated that the core used considerable power while doing read operations.

TABLE I. ROUGH POWER DISSIPATED WITH COMMUNICATION PHASES

PHASE	POWER DISSIPATED
IDLE	0.145280W
NOP	0.3W
COMPUTE	0.7W
READ	0.4W
WRITE	0.009999999983W

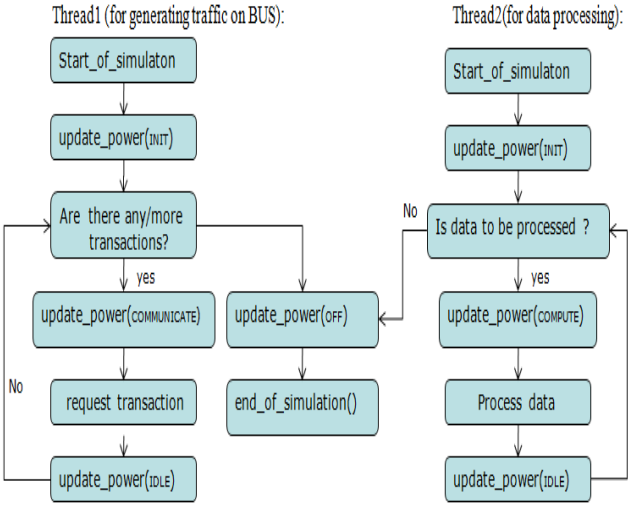

Fig. 13. Power Estimation Flow

VI. SIMULATION RESULTS ANALYSIS

The performance simulation step helps to identify and analyse major issues in the interconnect design, some of the experiences from interconnect design and performance simulations of VISTA SOC have been captured in this section.

a) Impact of "reducing latency on processor path"

As shown in Fig. 14 and Fig. 15, the small change of replacing the "Fconv" (frequency converter) with buffer gives an improvement of around 28 cycle's latency (14%) which is significant as it falls in the processor path. Note that, there was a constraint of "no frequency above 266 MHz at top level" in this SOC. If this constraint could be changed to higher frequencies, better latency numbers are possible.

b) Impact of "change in Topology"

The interconnect topology plays an important role in system performance and how fast we can arrive at the final working arbitration policies/tuning. The Node (Bus component) provides several kind of arbitration policies like, fixed priority by position, programmable fixed priority, Least recently used (LRU), bandwidth limiters etc. The LRU scheme ensures that all the initiators connected to the node get the "equal" share of bandwidth. "Equal" is not always "Fair" for a system like TV where there are numerous IP's with different bandwidth requirements. Selecting the right arbitration policy and tuning becomes a very tedious and time consuming step if the SOC interconnect is big. From number of iterations of performance simulations it was observed that it is better to identify IP's with similar bandwidth requirements and club them together (and put LRU arbitration scheme). Another major concern is the latencies for the Processors paths. Analysis and tuning of interconnect becomes very difficult if the packets coming from other initiators clients mixed with the processor transactions. There can be issues where packets from the processors cannot pass faster just because there are packets from other clients still waiting in interconnect. Tuning for arbitration at different levels in interconnects become very complicated. It was therefore observed that, for best results we should arbitrate the processor paths with other clients "as late as possible" as shown in Fig. 16 and Fig. 18.

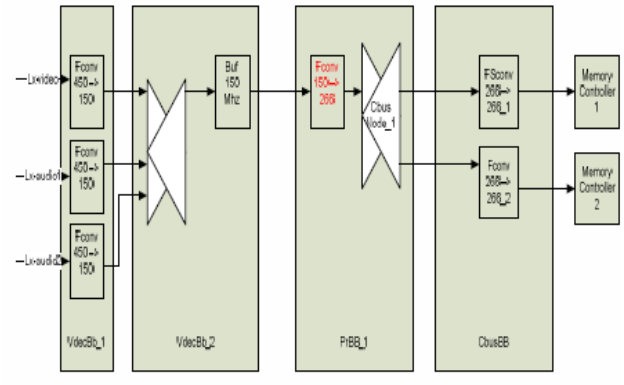

Fig. 14. Initial implementation(Total round trip latency i.e. for "req" to "r_eop" 188 cycles of 450 MHz clk (average))

c) Impact of "outstanding requests capability of IPs"

As the SOC complexity increases, the system latencies also grow proportionally. If an IP's capability to generate outstanding requests is limited, there is a direct impact on how much BW the IP can get to access the memory resources. If the IP is able to launch multiple outstanding requests into interconnect, the impact of high latencies are compensated. On running performance simulations for the SOC it was discovered that the 3D graphics IP was not hitting its bandwidth requirement. On further analysis it was observed that it is able to generate only one outstanding (each for RD and WR). As the IP generated LD/ST32's and the system average latency was around 200 cycles @ 200 MHz. So the maximum BW possible is 64 MB/s. Even if the Interconnect and DDR subsystem are able to sustain much larger bandwidths (in the range of GB/s); the IP could not utilize it effectively. Even if we tune/modify the interconnect/DDR controller and reduce the latency by half still the maximum achievable BW will be around 128 MB/s which is way behind what is expected. IP in the standalone could do much faster as the latency in the standalone environment is very low. But in a complex SOC for TV applications, it is impossible to achieve very low latency numbers for all IP's. So the only solution is to modify the IP to be able to launch more outstanding (increasing its internal pipe capability).

d) Impact of "space between requests"

IP's which have an average kind of BW requirement can have their requests spread out so that it is beneficial for the system BW. it is better if such IPs does not generate a "Peaky" b/w when all the blocks in the SOC are active. To achieve this, the requests generated by such IPs are "spaced" by a programmable parameter. Note that, this parameter need to change when the load on the DDR BW changes. The optimum value for this parameter can be derived only after running performance simulations for different use cases.

VII. FURTHER WORK

Migration of developed toolset for advanced interconnect architectures like ARM AMBA4 (ACE) is in progress with a completed automated flow. Efforts are being made to leverage the advantages of IPXACT standard to automate the flow from

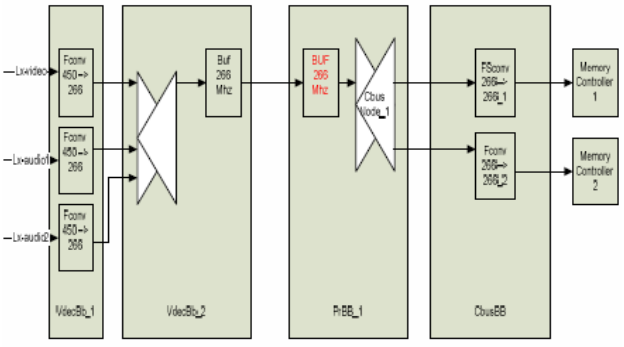

Fig. 15. corrected implementation (Total round trip latency i.e. for "req" to "r_eop" 160 cycles of 450 MHz clk (average))

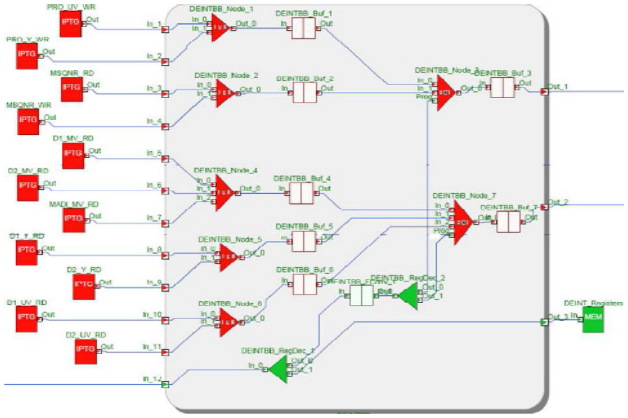

Fig. 16. Initial Design

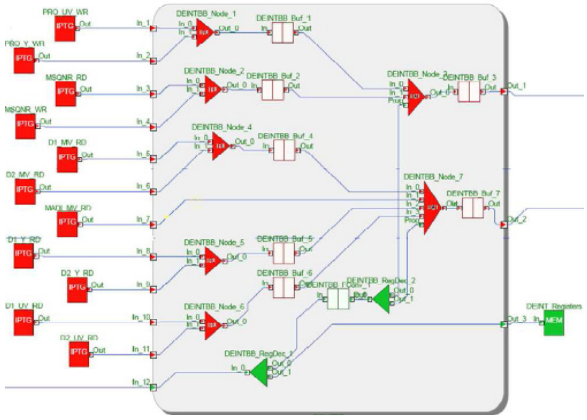

Fig. 18. Suggested Design

IP data sheet to behaviour models. Migrating to TLM 2.0 standard might be a further step to introduce interoperability of models with similar models from other vendors.

VIII. CONCLUSION

As the complexity of modern SOCs is increasing, the bandwidth estimation and analysis is becoming more and more complicated. Conventional methods of spreadsheet analysis are necessary but not sufficient to predict the system behaviour, more specifically in modern SOC which implements various kinds of functionalities which generate different kinds of traffic in the system. The proposed methodology for performance evaluation along with debug features embedded has several advantages, as shown in Fig. 17, over existing solutions like Emulation, RTL etc can be conveniently adopted in early SOC design phase.

Performance Parameter	Alternate solution(s)				Proposed solution
	Spread sheet analysis	High level C/C++	RTL	Emulation platform	Configurable BCA Models
Schedule impact	✓ Very Low	✓ Low	✗ High	✗ Very High	✓ Low
Overall Effort	✓ Low	✓ medium	✗ High	✗ High	✓ medium
Cost of tools	✓ Low	✓ Low	✓ medium	✗ Very High	✓ medium
Simulation time	✓ Low	✓ Low	✗ High	✓ Low	✓ medium
No of use cases	✓ Very high	✓ High	✓ medium	✓ medium	✓ High
Accuracy of results	✗ Very Low	✗ Very low	✓ High	✓ High	✓ High

Fig. 17. *Comparison of* proposed *methodology with existing flows*

IX. REFERENCES

[1] Gupta, R. and De Micheli, G., "Hardware/Software Co-design", *IEEE Proceedings*, Vol 85, No.3, March 1997, pp. 349-365

[2] K. Lahiri "Fast Performance Analysis of Bus-Based System-On-Chip Communication Architectures". Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD), November 1999, pp 566-572

[3] Pasricha, S. "Transaction level modeling of SoC with SystemC 2.0" in *Synopsys Users Group Conference (SNUG)*, 2002

[4] A. Clouard et al., "Towards Bridging the Precision Gap between SoC Transactional and Cycle Accurate Levels", in *Proc. of Design, Automation and Test in Europe Conf.*, 2002

[5] L. Cai and D. Gajski, "Transaction Level Modeling: An Overview," in *Proc. of Int'l Conf. Hardware/Software Codesign and System Synthesis"*, Newport Beach, CA, USA, 2003, pp. 19-24

[6] O. Ogawa et al., "A practical approach for bus architecture optimization at transaction level", in *Proc. of Design, Automation and Test in Europe Conf.*, 2003, pp. 176-181

[7] M. Caldari et al., "Transaction-level models for AMBA bus architecture using SystemC 2.0", in *Proc. of Design, Automation and Test in Europe Conf.*, 2003, pp. 26-31

[8] H. Jang et al., "High-Level System Modeling and Architecture Exploration with SystemC on a Network SoC: S3C2510 Case Study", in *Proc. of Design, Automation and Test in Europe Conf.*, 2004, pp. 538-543

[9] D.C. Black and J. Donovan, *SystemC: From theGround Up*, Springer-Verilag New York, Inc., 2005

[10] W. Klingauf. "Systematic Transaction Level Modeling of Embedded Systems with SystemC", in *Proc. of Design, Automation and Test in Europe Conf.*, 2005, pp. 566-567

[11] F. Ghenassia., *Transaction-Level Modeling with SystemC: TLM Concepts and Applications for Embedded Systems*. Springer, 2006

[12] Cornet, J., Maraninchi, F. and Maillet-Contoz, L., "A Method for the Efficient Development of Timed and Untimed Transaction-Level Models of Systems-on-Chip" in *Proc. Of Design, Automation and Test in Europe*, 2008, pp. 9-14

[13] Accellera Systems Initiative, "*SystemC"*, 2012, available at https://www.systemc.org