

Architecture Oriented Performance Optimizations for Bus Based System-on-Chip Designs Using TLM

Amit Agrawal, Nitin Gupta

IP Design Team
Coware India Pvt. Ltd.
NOIDA, India
{ amita, nitin }@coware.com

Abstract— The rapid evolution of product technologies is pushing more and more integration of discrete subsystems into a single chip. While integrated systems are boon for performance, size and speeds, however the quantitative visibility into performance behaviors is required to explore the design space. We have a methodology of cycle accurate Transaction Level Modeling (TLM) to generate the analysis data. This analysis data provides the required quantitative visibility. In this paper we discuss the way to utilize the analysis data in making design decisions for optimizing the architecture as well the software application. We describe how analysis data can be used to improve peripherals designs, hardware-software partitioning, interconnect configuration, optimizing the existent software for new hardware designs. A case study of AMBA Advanced eXtensible Interface (AXI) based systems is elaborated to explain the process of utilizing the analysis data for performance optimizations

Keywords - Architecture Exploration, TLM, AXI, Performance Analysis, SoC Platforms.

I. INTRODUCTION

With the advent of newer technology and tools to assist them, the System on Chip (SoC) designs are becoming more and more complex. Mapping of embedded applications onto single-chip multi-processor systems becomes a feasible and very interesting option. Design decisions take into consideration issues like costs, selection of peripherals and hardware/software partitioning besides the usual concerns of performance and design time (time-to-market) etc. One of the most important features for the new methodologies is the need to perform architecture exploration at higher levels of abstractions along with early software development [1].

SoC based multiprocessor architecture having heterogeneous peripherals and fast inter-connect are becoming very popular and hence, designers have to decide on the resources selection based on system performance against the application. This has increased the design-space manifolds. This problem is termed as Design Space Exploration (DSE) [4].

To address this Design Space Exploration problem, performance estimation at the system level is required. As SoC design components are sharing the ever increasing processing requirements there is considerable increase in communication

between these components. This makes bus performance to be a critical path in the whole system performance [7].

Shared bus architectures such as AMBA [10], AXI [9] and OCP [11] are, therefore, commonly employed in SoC designs. These bus architecture implementations are configurable to meet the application domain requirements. Hence, designers will be interested in having application specific bus configuration. To do this, designers simulate a proposed architecture based on empirical methods and basic principles of design like processor selection, cost considerations etc. The challenge is to improve and refine this architecture based on performance analysis as shown in Figure 1.

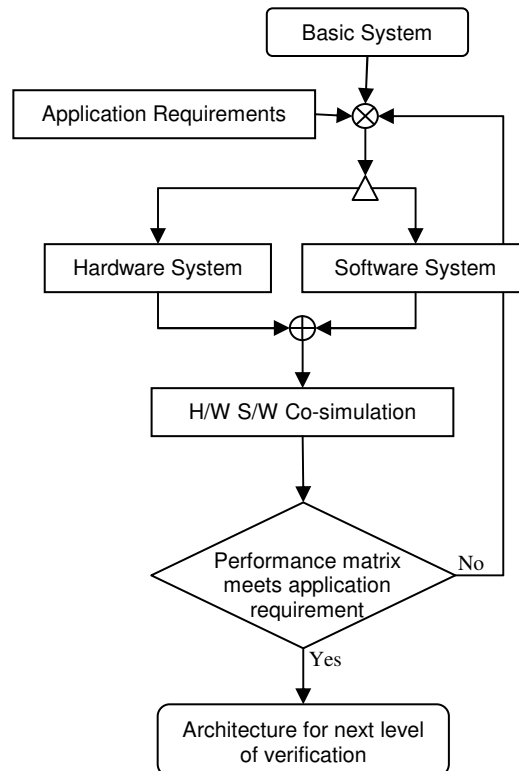


Figure 1. Architecture Exploration Methodology

We have a methodology of system simulation at TLM abstraction [8]. TLM is Transaction Level Modeling approach where system is simulated according to transaction flow. TLM abstraction is cycle accurate at transaction boundaries. This abstraction gives the advantage of both simulation speed of higher abstraction level and timing accuracy of lower abstraction level.[2][3] In this paper we propose the use of TLM methodology for the generation of performance matrices and its use in exploration of such complex multiprocessor bus based architectures. To underline the effectiveness of the methodology in architecture exploration, we undertook few typical SoC architectures based on AXI with multiple processing units and typical peripherals.

II. TLM METHODOLOGY

The CoWare TLM Methodology [8] applies to a class of designs that use bus-based inter-process communications typical to SoC architecture. The primary goal of Transaction Level Modeling is to achieve dramatically increased simulation speeds, while still offering enough accuracy for determining hardware response times. The main features of simulations based on CoWare’s TLM methodology are high speed simulation, cycle accuracy, reduced details and simple modeling, and support for HDL co-simulation.

Figure2 shows how the TLM Methodology can be used to simulate bus-based architecture. In such simulations, detailed signal handshaking is reduced to a series of generic events called ‘transfers’.

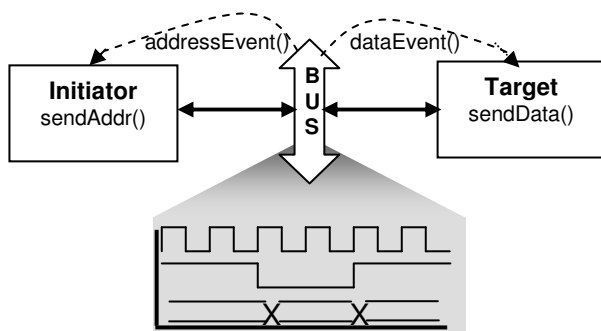


Figure 2. How TLM works

The Initiator and Target use an API to communicate via transfers on Bus model. The bus model keeps track of timing, and events on the bus can be used to trigger action on peripherals. Transfers reduce communication details to a small number of events. An API is used to control transfers. This includes initiating transfers, and setting transfer attributes. Pipelines can be handled in this approach by the bus. Transfer attributes can be presented to the bus at one time and they will be pipelined by the bus. One of the major advantages of using TLM methodology is increase in speed of simulation. In a typical modeling style, processes in each block are driven by a clock. This causes processes to be evaluated with each clock, even if there is no change of state thus having a penalty on simulation performance. At TLM level, bus is triggered on clock and rest of the system is triggered by bus, processes can be sensitive to transfer events on the bus, so processes are only

evaluated when there is a relevant event. This reduces the number of events that need to be evaluated with each tick of the clock. Another key feature of TLM is the separation of communication and behavior in a device model providing flexibility in modeling. Communication can be described in a wide range of fashion, from high level messages to detailed signal level handshakes without impacting the behavior description. Behavior can be described algorithmically, without the burden of the handshaking and control logic associated with bus communication

III. PERFORMANCE METRICS

The main aspect of architecture exploration is to assess architecture against system performance goals. To accomplish this it is required to get overall system behavior quantitatively. These system indices at various stages are mapped into performance metrics. During architecture explorations following performance metrics are mainly taken into consideration.

1. Overall system response and throughput.
2. System bottlenecks.
3. Bus contention.
4. Software performance.
5. Peripheral utilization.
6. Processor loading.

The list could include many more values but typically these are of concerns to most of the system designers. From the CoWare’s TLM simulation we generate the performance metrics as analysis data. The bus models are instrumented in a way to generate this analysis data.

For this study, we are taking the following analysis data in consideration:

A. Latency Statistics

Various latencies in the system impact overall performance of the architecture and hence, performance improvement can be achieved by reducing these latencies. Figure3 shows the value of various latencies of a typical AXI system at different cycles.

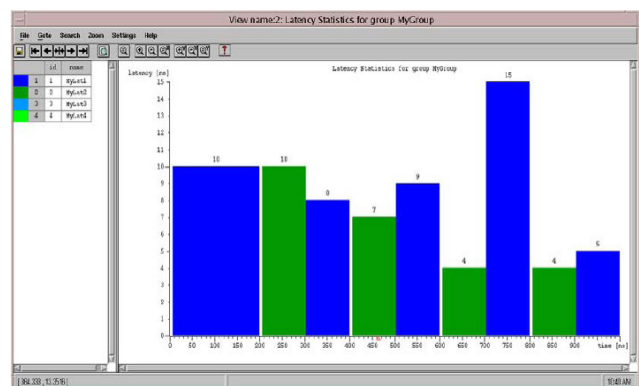


Figure 3. Latency Statistics

Latency statistics can be used to re-configure the shared-bus architecture according to the performance goal

B. Bus contention statistics

Bus contention statistics provides the analysis data in terms of number of masters waiting for bus grant at a particular node. It also provides wait time for each master while requesting bus. This again helps in redefining the interconnect model so as to reduce the wait-time for each master as well as to reduce number of masters blocked waiting for bus at any given time.

C. Transaction Statistics

The transaction statistics shows the utilization of masters, slaves, and bus. Utilization is defined as the percentage of the time the masters and slaves are busy handling a transaction. The master is considered busy the moment the transaction starts until the moment the transaction ends. The slave is considered busy from the moment the transaction reaches the slave until the moment the transaction ends. The bus is considered utilized if at least one master is busy. Figure4 shows the utilization statistics of a system with three masters at various intervals.

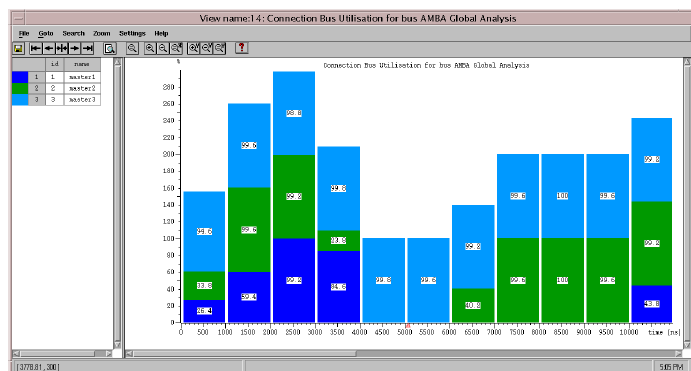


Figure 4. Transaction Statistics

This transaction utilization data can be used to optimize the peripherals and improve processor loading. This can also be used to reconfigure the interconnect parameters to improve Bus Utilization factor.

D. Throughput Statistics

Throughput statistics shows the data transmission rate at each node of the inter-connect in a shared bus architecture. This also shows the overall system throughput. Throughput statistics can be used to improve the software run-time performance. Software can be enhanced to utilize bus resource effectively thereby increasing throughput.

IV. ARCHITECTURE OTIMIZATION METHODS

Here we present a study of some architecture optimizations that we have performed using the TLM modeling approach and the generated analysis data. We modeled AXI based multiprocessor SoC architectures. Here, we describe architecture evaluation process by taking sample master and slave devices and AXI bus libraries. The masters are SystemC

implementation of file readers that fire transactions on the bus based on inputs provided in the stimulus file. The stimulus file has the provision for setting various transaction attributes and various use cases of the architecture. SystemC models of Memory with configuration to set various latencies are used as slaves. Various optimizations that are possible on the models we have created are described below. They have typical SoC design scenarios that the designers are interested in.

A. Peripheral Optimizations

Standard architectural decisions are made at the early designing phase of the system about the peripherals and their topology. It is important to get information about what peripheral configurations can improve system performance. [1]

To simulate the situation, we created two architectures Mod1 and Mod2 with a single AXI memory that has a single port in Mod1 and dual port in Mod2 as shown in Figure5. The masters are performing incrementing burst mode 128-bit data write transactions on the memory and are connected to the input stage node of the AXI bus. The memory model acts as slave and is connected to the output stage of AXI node. The memory has a write latency of 2 cycles and burst write latency of 1 cycle. Both the masters are accessing memory in different pages.

It is a typical DMA mode memory operation which is very common in bus-based SoC systems. The simulation is repeated with read transactions for the same memory locations.

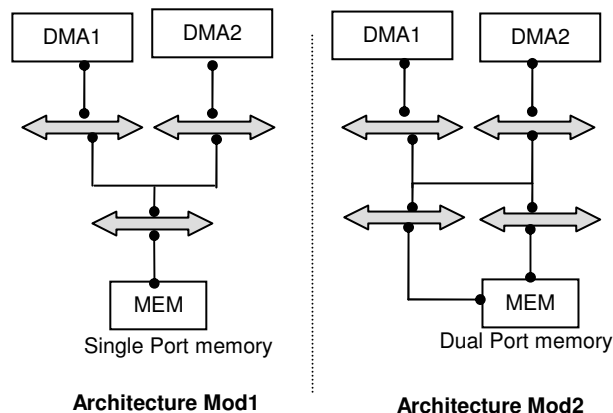


Figure 5. Peripheral Optimization Models

TABLE I. TOTAL TIME OF SIMULATION

Transaction	Mod1	Mod2
Write (100)	32980ns	32050ns
Read (100)	32050ns	31630ns

TABLE I shows the time taken by the two simulation models. From these values, it could be inferred that there is a gain in performance in Mod2. The gain is limited in AXI

because AXI protocol does not let the slave block the bus for other master.

This data about peripheral performance can be obtained by simple changing of peripheral settings in minimal time, thus helping the designer to take decision based on accurate quantitative results. It reduces the costly and time consuming change feedbacks from the later design stages thus improving the overall effectiveness of designers.

B. Processor Loading (Selection).

To leverage system performance, designers are putting in more and more processing units in their designs and one of the major questions that are being asked is how fast the processing needs to be done. The Bus contention statistics and the master utilization statistics provide valuable data about selecting the kind of processor. Still, a quantitative comparison can be done by getting accurate system behavior to take decisions on processor speeds.

We created a platform having 20 masters connected to 20 input-stage nodes of AXI and are firing write data transactions on two different memory slaves. The two memory slaves are connected to the output-stage of the AXI nodes and have typical latency values. The requirement here is to find out the ideal processing power of each of these 20 masters in order to get maximum system performance, thus reducing the overall system cost.

The bus contention statistics show that the bus is fully utilized when all these masters start firing transactions. This causes the processors to wait for the bus grant and hence they are under-utilized. In our simulation environment, the master stimulus was generated by putting idle cycles for each master to get the required throughput.

Our model had bus cycle as 200Mhz and width as 128 bits. A master has to be operated at 400Mbps. Hence, it must remain on this thread for 64 clock cycles. It has to complete its task for this thread in these cycles and if there are remaining cycles, it should stay idle. Similar calculation can be done for all the masters to get their idle cycle count based on the required data fetch from the bus.

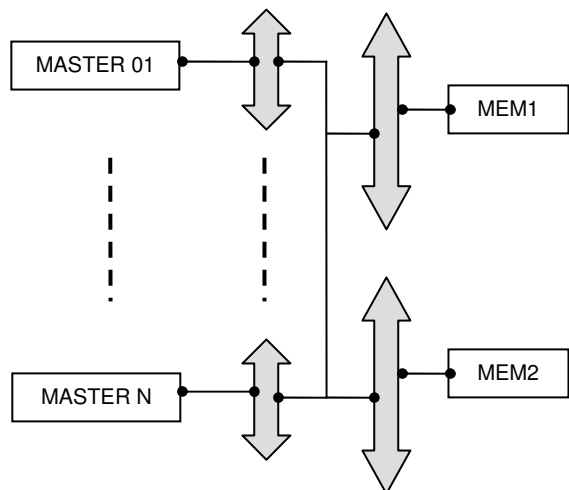


Figure 6. N Masters Connected to 2 slaves.

In this system, we could calculate the required processor data bandwidth for a full cross bar AXI system running a typical embedded application. This scenario takes into consideration that the bus contention is optimized to the best level and then the processors are to be selected.

Another interesting feature that can be noted here is that the master idle cycles give information about the masters that do not have bus-communication processes and hence can be loaded with computational tasks. Also, if a master is granted bus in its idle cycles, then it can be loaded with bus-communication tasks so that better system throughput can be achieved otherwise that particular cycle is wasted.

Processor loading can also be measured on the basis of inter-connect capacity and the peripheral capacity to perform I/O operations.

C. Migration of InterConnect Models.

Typical bus based SoC architectures use various interconnect protocols like AMBA, AXI, CoreConnect etc. Designers need to make decisions on the selection of the inter-connect based on system performance against cost. Here, analysis data can be used to generate quantitative system performance information.

We undertook a simulation of migrating a system based on AMBA multilayer bus to AXI bus. AXI is similar to AMBA multistage, however the AXI also provides parallelism between read and write transactions. For the particular application and peripherals, analysis data can be generated for both AMBA as well as AXI based systems.

We created platforms for the performance simulation of the AXI systems with 4 masters and 3 memory slaves. To simulate particular I/O requirement, each of the master was firing word write transaction on to the same slave at a particular rate. The bus throughput, resource utilization at each node and over all system performance statistics is gathered for the particular interconnect configuration. These values could be plotted against the I/O requirements.

The cost consideration for the interconnect selection could be the factor of

- Interconnect Configuration
- I/O requirements

Graphs could be plotted for the Interconnect statistics against cost considerations to take decision for moving to AXI interconnect model for a particular application requirements.

We generated values for the bus throughput against the number of writes per second in a standard AXI based platform and they are plotted in Figure8. As expected, the bus throughput increased in AXI when the writes per second increased. The graph also depicts that as the number of writes per second increases, a threshold is reached and the bus throughput declines. This decline is due to the increased delay in arbitration due to multiple writes pending in the system. System designers can get an estimate of the writes that the application should be performing to get maximum system performance.

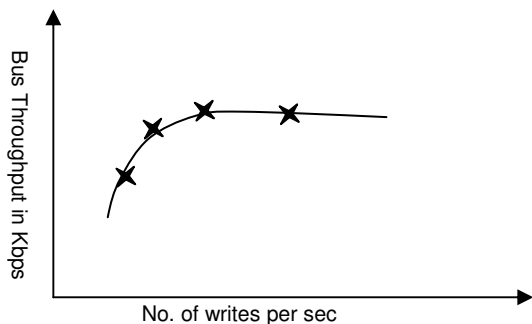


Figure 7. AXI Performance.

D. InterConnect Configurations.

The most obvious dimension of the design space exploration is the bus architecture topology itself with various permutations of number of bus nodes, dataflow handling and scheduling schemes within the bus itself.

Shared pipelined bus architectures like AXI could be configured for various parameters like

1. Number of bus nodes.
2. Priority of a particular bus node.
3. Arbitration scheme.
4. Queue depth of pending transactions i.e. maximum number of simultaneous transaction pending at a particular time.
5. Queue depth of pending particular transaction stages for example maximum number of the write data transfers pending at particular slave.

The TLM methodology could be used to generate optimum interconnect configuration for the particular application. The simulation platforms can be configured for the required interconnect parameters to generate the analysis data. The cost comparison of the analysis data could be done to arrive at optimum interconnect configuration.

We created a typical bus based system with fourteen masters firing word write transaction in burst mode to 3 slaves at different memory locations. The bus was configured with the write issuing capability $N \{1..16\}$. The system performance improved as the write issuing capability is increased but it stabilizes after $N=10$. Increasing the write issuing capability means adding hardware and, hence, increases the cost of the system. Similar experiments can be performed on variation of other bus configuration parameters.

It helped estimating the bus configuration against the performance–cost trade-off.

E. Software Optimizations

Hardware – Software co-design and co-verification has become a primary objective of complex SoCs to reduce the time to market. Designers need to create application specific hardware. Software systems having I/O critical sections based on bus communications need to be mapped to various I/O

processing blocks in the system. In systems with multiple blocks having different priorities, it is a challenge to map the software sections to specific hardware blocks. In the past, various studies [5] [6] have tried to split the software into sections that can be mapped to multi-core architectures. Here, we are proposing a similar approach based on the analysis data generated after mapping the software’s I/O critical sections to various processing blocks.

We analyzed a typical embedded application (MPEG-IV Decoder) with software profiling tools to obtain the I/O critical sections at run time. We created a simple platform with an ARM core having the highest priority along with two DMA having lower priorities. The software code is kept in Mem1 and the data is in Mem2 as shown in Figure9. The processed data is written in Mem2.

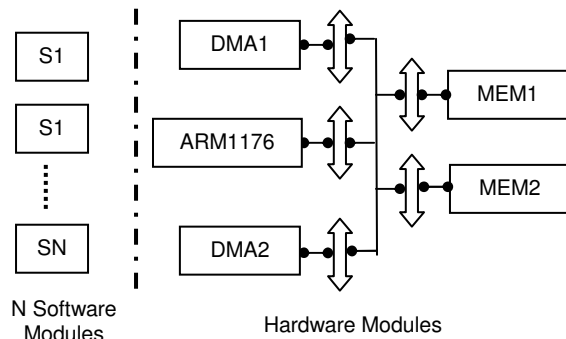


Figure 8. Mapping Software Blocks to Hardware.

The Software analysis against the bus contention graph gives information about what piece of code is waiting for the bus grant. Such code snippets need to be optimized and in some cases they may be allocated to higher priority masters in order to improve overall system throughput.

V. CONCLUSION

The proposed architectural exploration technique can help the designers in analyzing hardware models against the required performance and cost goals. Cycle accurate simulation at transaction level (TLM) is faster and very appropriate for such explorations. The time to design systems in this framework is much lesser than that in traditional design methodologies and the analysis results generated through this methodology are as accurate as any low level simulation.

VI. ACKNOWLEDGMENT

We would like to thank Mr. Sanjay Chakravarty for the time and efforts in helping us to perform this study. We would like to thank for the co-operation and help extended to us in the study by the IP Team of CoWare India.

VII. REFERENCES

- 1 Osamu Ogawa et al, "A Practical Approach for Bus Architecture Optimization at Transaction Level", Proceedings of DATE 2003
- 2 Sudeep Pasricha, Nikhil Dutt, Mohamed Ben-Romdhane, "Rapid Exploration of Bus-based Communication Architectures at the CCATB Abstraction" In CECS Technical Report 04-11, May 2004
- 3 M. Caldari et al, "Transaction-Level Models for AMBA Bus Architecture using SystemC 2.0", Proceedings of DATE 2003.
- 4 A. Baghdadi, N-E. Zergainoh, W. Cesario, T. Roudier, and A. Jerraya, "Design Space Exploration for Hardware/Software Codesign of Multiprocessor Systems". In Intl. Workshop on Rapid System Prototyping, 2000.
- 5 Dr. Akash R. Deshpande, "Software Performance Considerations for multi-core Processors", Teja Technologies White Paper.
- 6 Venkataramani, G., W. Najjar, F. Kurdahi, N. Bagherzadeh, W. Bohm. "A Compiler Framework for Mapping Applications to a Coarse-grained Reconfigurable Computer Architecture." Conf. on Compiler, Architecture and Synthesis for Embedded Systems (CASES 2001), 2001.
- 7 Holzer M, Knerr B, Belanović P, Rupp M, "Faster Complex SoC Design by Virtual Prototyping", Proceedings of the International
- 8 Conference on Cybernetics and Information Technologies, Systems and Applications (CITSA 2004), pp. 305-309, Orlando, 2004
- 9 Pete Hardee, CoWare Transaction Level Modeling Methodology white paper http://www.coware.com/PDF/CONVERGENSC_TLM.PDF
- 10 AMBA AXI Specification <http://www.arm.com/armtech/AXI>
- 11 ARM AMBA Specification Document http://www.arm.com/products/solutions/AMBA_Spec.html
- 12 IBM CoreConnect bus architecture <http://www-03.ibm.com/chips/products/coreconnect/>
- 13 T. Grötker, S. Liao, G. Martin, S. Swan, "System Design with SystemC." Kluwer Academic Publishers, 2002.