

# A Methodology for Performance Analysis of Network-on-Chip Architectures for Video SoC

Krishnan Srinivasan, Sonics Inc  
Erno Salminen, Tampere University of Technology

## **ABSTRACT**

High end System-on-Chip (SoC) architectures consist of tens of heterogeneous processing engines and memory units. The traffic profiles from each of the processing engines vary with respect to their bandwidth and latency requirements. An interconnection network (or Network-on-Chip (NoC)) provides the infrastructure for these processing engines and memory units to communicate. With increasing system complexity, the performance of the system is increasingly determined by the ability of the interconnection network to provide sustained data bandwidth to the engines, and at the same time, meet the real time performance goals of latency sensitive traffic. Hence, the interconnection network must be analyzed together with the whole system and not in isolation. In this paper, we present a methodology for performance analysis of the interconnection network, with focus on video and multimedia benchmarking. We describe a typical video decoder based SoC system, and describe the traffic profiles for each of the processing engines. We also provide performance analysis measures of interest in a video decoder based SoC.

## **1. INTRODUCTION**

A System-on-Chip (SoC) consists of several processing engines integrated onto the same chip. The traffic flow from each of the processing engines can have different performance requirements [1]. For example, a processor whose traffic to the external DRAM is dominated by cache-line fills requires that the latency of the traffic is minimized. On the other hand, traffic flow from a video decoder engine requires that the traffic is serviced with the low jitter (instead of latency per transaction), such that the amount of buffering within the engine can be minimized.

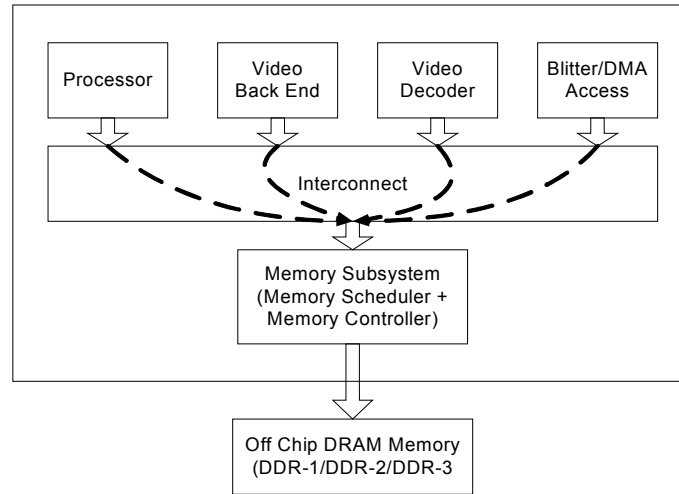
Figure 1 depicts a typical SoC architecture for HDTV (high definition TV). For clarity, only the major blocks are included. The initiators are connected to the interconnection network by means of a communication interface, such as the Open Core Protocol (OCP) interface. The majority of traffic flows from the different processing engines (henceforth called Masters or Initiators) converge into the DRAM memory. In other words, the system operates as shared memory architecture and all bandwidth intensive communication takes place through the external DRAM architecture. Therefore, the low bandwidth traffic not targeted at the DRAM architecture is not modeled. The memory subsystem consists of a memory scheduler, a memory controller, and the off-chip DRAM. The memory scheduler arbitrates among multiple traffic flows to generate requests to the controller, which are then issued to the DRAM.

With each passing generation, the complexity of the system is increasing. The major artifacts of an increasingly complex system are:

- i) The number of masters is increasing from low tens, to the high tens or even hundreds.
- ii) The bandwidth at the individual masters is increasing.

Finally, in order to sustain the increasing bandwidth requirements, the number of memory channels is increasing from one to two to four and eight in the future. Therefore, a system with fifty initiators communicating with four DRAM channels (where each DRAM channel has its own NoC terminal) will introduce two hundred traffic flows, each with its own bandwidth and latency requirement.

The increasing complexity of the SoC architectures has led directly to the increase in complexity of the interconnection architecture. The interconnection architecture has to solve a multi-objective problem of satisfying the bandwidth requirements, satisfying the latency requirements, minimizing the gate count in the design, and minimize power consumption. Such complex interconnection architectures have brought about a need for structured performance analysis [2].

**Figure 1: SoC system with DRAM**

More and more SoC designers are moving toward an intellectual property (IP) and platform based design methodology [3]. A typical SoC may be composed of a processor from a processor vendor, some locally designed blocks, an interconnect and memory subsystem from an interconnect vendor, and the DRAM memory subsystem from a DRAM vendor. Marculescu et al. [6] identify several outstanding problems for future generation NoC design. For the interconnect vendors, the analysis of the interconnect with a system view has become a major problem. Often, the interconnect vendors do not have the access to complex initiators that constitute the SoC. Hence, they are either forced to design with the worst case scenario in mind, which can lead to unnecessary redundancy. The other option, which is to design the interconnection architecture based on random flows can force re-architecting the NoC late in the system design cycle, as it may not satisfy the SoC's requirements.

In this paper, we describe a methodology for the performance analysis of the interconnection architecture, with focus on SoC architectures for video applications such as HDTV, set-top box etc. Our approach is to describe a typical video SoC, and provide a set of traffic profiles for the initiators that constitute the SoC. With the traffic profiles as basic building blocks, video SoC benchmarks of varying complexity can be constructed by varying the number of initiators and the amount of bandwidth per initiator.

We provide traffic profiles for various initiators such that they can be modeled in an abstract manner, thus minimizing the development effort. In addition to the traffic profiles, we also provide essential measurement schemes designed for video benchmarking. With the recommendations of this paper, the interconnect designer can model traffic flows that mimic typical video systems and thus, can make the necessary architectural decisions early in the design cycle.

In the paper, we use the semantics of the OCP to explain the methodology. However, the methodology itself is not dependent upon the protocol. The paper is organized as follows. In Section 2, we describe the essentials to describe a video SoC. In Section 3, we describe the traffic profiles for the different traffic generators in the system. In Section 4, we describe the overall approach to generate a video SoC benchmark. In Section 5, we present details about the performance analysis of video SoC systems. Finally, in Section 0, we conclude the paper.

## 2. Description of Video SoC Application

A typical video SoC consists of heterogeneous processing engines that primarily communicate with the external shared DRAM memory. The processing engines operate in a fully pipelined fashion, and for the purposes of the performance analysis of the interconnect, they can be thought of as operating as independent initiators. The major processing engines in a video SoC are Processors, Transport engines, Video decoders, Audio processing units, Graphics engines, Display processors and a peripheral display cluster. Hence, there are 7 distinct types of processing engines. The memory subsystem primarily consists of the DRAM memory, and can consist of one, two, four or eight DRAMs.

With these units, a SoC can be put together in a similar manner described in Figure 1, where all major communication takes place through the shared DRAM architecture. In order to describe the system, the system designer needs to consider the following parameters:

1. The number of initiators in the SoC:

- a. We note that a video SoC will be composed of at least one component instance of each initiator type, where an initiator type could be a Processor, Transport engine, Video decoder, Audio processing unit, Graphic engine, Display processor or a Peripheral unit. It is however, possible to specify multiple such units. For example, the SoC may be composed of two processors, two video decoders and one of each of the other initiator type.
  - b. OCP supports optional per-thread flow control, enabling the delivery of fully non-blocking interfaces in a manner analogous to virtual channels with virtual flit flow control. Within a thread, transactions may have independent ordering tags. For each initiator, the designer must specify the number of OCP threads and number of OCP tags per thread in the corresponding initiator OCP interface.
2. The performance requirements of each initiator
    - a. The performance requirements of each initiator can be specified based on
      - i. worst case latency,
      - ii. average latency
      - iii. the required sustained bandwidth.
    - b. The requirement of each initiator type will be different.
  3. The DRAM memory architecture
    - a. The DRAM architecture could be composed of DDR-1, DDR-2, DDR-3 or XDR (dual data rate (DDR), extreme data rate (XDR)). For the DRAM architecture, the type of part, the number of parts, the DRAM burstlength, and the operating frequency of the DRAM bus is specified. Further, the number of DRAM channels in the design is also specified. Each channel can be accessed independently of others and each channel has its own DRAM scheduler and controller. On the other hand, multiple DRAM parts are used within a DRAM channel to widen the DRAM data width and thus, achieve the required bandwidth from the DRAM channel.
    - b. Note that the memory subsystem is defined by the evaluator. Hence, the evaluator must document access latencies of the DRAM memories.
  4. The width of the communication interfaces.
    - a. The system designer specifies the width of the communication interfaces between the initiators and the interconnect, and the interconnect and the memory subsystem.
  5. The operating frequency
    - a. The system designer should specify the operating frequency of the communication interfaces. In this version, we do not explicitly define clock domain crossing requirements. We assume that the NoC handles the domain crossing, wherever required.

The interconnection architecture should satisfy the performance requirements of each initiator, subject to the other design parameters provided by the system designer. In addition, it should be able to provide a measure of the number of gates that are consumed from an overall system perspective. For example, choosing a particular interconnection architecture can result in the minimization of the buffering in one of the initiators, while increasing the gate count of the network itself. Hence, the overall evaluation must take into account the relative advantages and disadvantages taking the system as a whole, instead of evaluating the interconnect in isolation.

### 3. Traffic Profile Generation

This section, describes the traffic profile generation for each initiator. The traffic profiles are generated in three levels of hierarchy:

1. The designer instantiates an abstract SoC system by specifying the number of initiators and their type, total bandwidth requirement of the system, the number of DRAM channels, the details about the DRAM architectures, and the details of the OCP interfaces.
2. The designer specifies details about each initiator type. This includes the type of accesses, locality of addresses etc.

3. Finally, the designer specifies the details about each initiator. For example, each initiator specifies the number of outstanding transactions per thread that the initiator can support.

In the remainder of the section, we describe the hierarchy level-1 specifications, followed by the traffic profile for each initiator type. In describing the traffic profile, we also describe the set of parameters associated with each initiator type at the second level of hierarchy, and initiator in the third level of hierarchy.

### 3.1 Hierarchy Level-1 Specifications

The level-1 specifications include the details about the entire system, without focusing on the individual initiators. The system designer provides the following parameters:

1. The number of initiators of each
2. The total bandwidth requirement of the system, which would be the sum of the individual bandwidth requirements at each initiator
3. The number of DRAM channels
4. The frequency of each DRAM bus
5. The number of parts in each DRAM
6. The type of part in each DRAM
7. The DRAM burstlength for each channel
8. The data width of each initiator OCP interface
9. The operating frequency of each initiator OCP interface
10. The number of threads in each initiator OCP interface
11. The number of tags per thread in each initiator OCP interface

### 3.2 Processor Traffic Profile

The CPU deals with cache-line sized chunks primarily. Cache line sizes are usually 32 bytes or 64 bytes. A dirty line replacement results in a Writeback (WB). The cache line fetch and writeback operations manifest as a read or write respectively, of burst-length depending on cache line size and OCP interface data width. Finally, depending on the cache organization, there can be separate instruction cache. If an instruction cache is present, then there are only read misses, since writebacks are absent.

In this paper, we abstract cache hierarchy away by expressing the miss rate coming to the interface as “*misses per instruction*” (MPI) rather than the misses per access. Therefore, we make the traffic profile insensitive to the cache hierarchy of the system. Another advantage of the MPI measure is that it is, in part, an indicator of the CPI (cycles per instruction) performance measure. For our purposes, we can use the misses per cycle (MPC) measure instead of the MPI ( $MPC = MPI/CPI$ ).

#### 3.2.1 Processor Characteristics:

1. Cache line size (instruction and data cache): (default 32B)
2. Read misses per cycle: 0.01 – 0.001
  - a. Therefore, a read miss should be issued every 100 – 1000 cycles. The misses occur irrespective of the completion of previous misses until the initiator cannot support anymore outstanding transactions. When there are already maximum number outstanding transactions, initiator must wait until at least one of them completes before continuing. This means that the data flow may be momentarily stopped.
3. Write misses per cycle (0.2 to 0.3) \* *Read misses per cycle* for data cache
  - a. 0 for instruction caches

4. Temporal distribution of accesses:
  - a. Uniform distribution (this is generated from a combination of read misses and writebacks)
  - b. With separate instruction cache, the misses are typically bursty because of high temporal and spatial locality but the MPC itself is much lower (0.001 – 0.0001). Burstiness can be modeled, for example, with the b-model[4].
5. Spatial distribution of accesses (i.e., address generation)
  - a. Random address distribution
6. Bandwidth requirement
  - a. Bandwidth requirement of all CPUs together is about 15 % of the total bandwidth requirement of the SoC.

Based on the processor characteristics, the following hierarchy-level-2 and hierarchy-level-3 parameters can be derived.

#### ***Level-2 Parameters***

1. Distribution of bandwidth among multiple processors
  - a. For example, if there are 2 processors in the system, the designer may choose to assign 1/3 of the processor bandwidth to processor-1, and the remaining 2/3 to processor-2. Hence, the bandwidth on processor-1 will be  $1/3 * 15\%$  of the total system bandwidth, which equals 5% of the total system bandwidth. The remaining 10% of the total system bandwidth will be assigned to processor-2.

#### ***Level-3 Parameters***

1. Cache line size 32 bytes or 64 bytes
2. Read miss and Write-back MPC per thread of the initiator OCP.
  - a. The value should be within the specified range, and should meet the bandwidth requirement at the initiator.
3. A value to indicate if instruction cache is available
  - a. If instruction cache is available, 1.5 % of the processor's traffic will be random reads. This is in addition to the above mentioned read miss every 100-1000 cycles
  - b. Without cache, no random reads
4. Outstanding transactions per thread. We measure the outstanding transactions in bytes. When this value is set to  $N$ , the processor generates at most  $N$  bytes of transactions per thread before stalling on that thread to receive the response to the first of the  $N$  transactions. When the CPU is stalled, it will not generate any further misses (i.e traffic).
5. Average and worst case latency that the processor can tolerate
6. The percentage of traffic going to each thread, and with each tag ID, within each thread, and the total number of transactions.
  - a. This value represents the distribution of traffic on each thread/tag.

### **3.3 Display Processor Profile**

The display processor performs a variety of functions such as multi-field adaptive interlacing, motion interpolation, color correction, scaling, and noise reduction. This manifests in typically long bursts with spatial locality. The traffic typically consists of alternating periods of high activity, and no activity. During the period of activity, the initiator generates transactions at random intervals, such that the bandwidth during the active period is twice that of its average bandwidth. The traffic generator stalls if it reaches the maximum number of outstanding transactions.

### 3.3.1 Display Engine Characteristics:

1. Long bursts: burstlength of 128 – 384 bytes
2. Read to write ratio: ~2 – 3
3. Reads and writes go to different memory banks
4. Read addresses are correlated (spatial locality)
  - When addresses are spatially correlated, the burst has incrementing addressing over large windows, for example 512 bytes. This can be modeled by a Weighted random selection of address. For example, start at a random address, and have incremental addresses for 512 bytes. An example address sequence would be: (x = rand(), x+1 .... x+ 512).
5. Write addresses are correlated (spatial locality) similarly to read addresses.
6. Reads and writes are randomly distributed temporally, but they must satisfy the bandwidth requirement
7. The bandwidth is 40 to 50 % of the total system bandwidth.

From the characteristics, the following level-2 and level-3 parameters can be derived:

#### **Level-2 Parameters**

1. Distribution of traffic among the multiple display engines
2. Amount of traffic for the display engine. It should be between 40 % and 50 % of total soc traffic.
3. Length of the one period of activity as a percentage of the total simulation, which is a measure of the number of frames being processed in one simulation

#### **Level-3 parameters**

1. Burst length of the initiator
2. Number of outstanding transactions per thread
3. Number of OCP threads and OCP tags per thread on the initiator OCP interface
4. The percentage of traffic going to each thread, and with each tag ID, within each thread, and the total number of transactions.

## **3.4 Video Decoder Profile**

The video decoder performs motion compensation/estimation, De-blocking filter, DCT transform, De-Quantization and Entropy Encoding. The operations are often performed on blocks of images, thus resulting in several OCP BLCK bursts. Each row in the BLCK burst is typically 16 to 64 bytes long. The number of rows is generally between 2 and 16, with the total size of the burst being 128 bytes to 384 bytes. Similar to the display processor, this unit also consists of alternating periods of high activity and quiet period. The period of activity has traffic flow which is twice the programmed bandwidth. Within an active period, the traffic is constrained random.

### 3.4.1 Video Decoder Characteristics:

1. BLCK bursts: burstlength of 16 – 64 bytes per row, 2 to 16 rows, and 128 to 384 bytes of total burst size
2. Read to write ratio: ~2 – 3
3. Reads and writes go to different memory banks
4. Read addresses per row are correlated (spatial locality)
5. Write addresses per row are correlated (spatial locality)
6. Two rows in a burst are separated by an address offset that is a power of 2. For example, 0x1000.
7. Reads and writes are randomly distributed temporally, but they must satisfy the bandwidth requirement
8. The bandwidth is 20 to 30 % of the total system bandwidth

From the characteristics, the following level-2 and level-3 parameters can be derived:

***Level-2 Parameters***

4. Distribution of traffic among the multiple decoder engines
5. Amount of traffic should be between 20 and 30 %
6. Length of the one period of activity as a percentage of the total simulation, which is a measure of the number of frames being processed in one simulation

***Level-3 parameters***

5. Burst length of the initiator
6. Number of outstanding transactions per thread
7. The percentage of traffic going to each thread, and with each tag ID, within each thread, and the total number of transactions.

**3.5 Graphics Processor/ Blitter Profile**

The graphics processor unit performs Color fill, Rectangular copy, Color space conversion, Alpha blending, Gamma correction, Resizing and Clipping. The burst length of the transactions are typically large, about 256 bytes is size. Similar to the display processor, this unit also consists of alternating periods of high activity and quiet period. Within an active period, the traffic is constrained random. The period of activity has traffic flow which is twice the programmed bandwidth

***3.5.1 Graphics Processor Characteristics:***

1. Burstlength: 128-256 bytes
2. Read to write ratio: ~2 – 3
3. Reads and writes go to different memory banks
4. Read addresses are correlated (spatial locality)
5. Write addresses are correlated (spatial locality)
6. Reads and writes are randomly distributed temporally, but they must satisfy the bandwidth requirement
7. The bandwidth is 10 to 15 % of the total system bandwidth.

***Level-2 Parameters***

1. Distribution of traffic among the multiple decoder engines
2. Amount of traffic should be between 10 and 15 %
3. Length of the one period of activity as a percentage of the total simulation, which is a measure of the number of frames being processed in one simulation

***Level-3 parameters***

1. Burst length of the initiator
2. Number of outstanding transactions per thread
3. Number of OCP threads and OCP tags per thread on the initiator OCP interface
4. The percentage of traffic going to each thread, and with each tag ID, within each thread, and the total number of transactions.

### 3.6 Audio Processor, Transport and Peripheral Cluster

The audio processing unit and the peripheral clusters are low activity initiators, and make up the remaining of the traffic in the SoC system. They can be randomly generated within the available bandwidth. As a first approximation, these traffic profiles have burst sizes of 8 bytes with a read to write ratio of 2:1.

## 4. Overall Approach

The overall approach to performance benchmarking of NoCs for video SoC is described in the following steps:

1. The level-1 parameters regarding the overall system are specified.
2. For each initiator type, the level-2 parameters are described.
3. For each initiator, the level-3 parameters are specified
4. A derivation algorithm should read the parameters of the system, and generate traffic for each initiator type, based on the characteristics of the initiator, and the level-1/level-2/level-3 parameters.

### 4.1 System-Configuration

The following enumeration describes the range of values to be used to specify each parameter in the system. Based on this, multiple designs can be generated, which can be used for performance analysis.

1. Number of initiators of each type : 1 to 4
2. Number of threads per initiator 1 to 4
3. Number of tags per thread 1- 8
4. Total bandwidth of the system: 1.3 GBytes/sec – 10.5 GBytes/sec
  - a. The 1.3 GBytes/sec corresponds to roughly to an MPEG-2 HD video, about 2.5 GBytes/sec for MPEG-2 with judder removal and motion compensation, 4.6 GBytes/sec for H264 decoder, and 10.5 GBytes/sec for dual H264 decoder format [5]
  - b. As a reminder, the bandwidth is divided among the initiator classes as follows:
    - i. CPU 15%
    - ii. Display Processors 40-50%
    - iii. Video Decoder 20-30%
    - iv. Graphics Processors 10-15%
    - v. Other Initiators less than 5%
  - c. The activity of each initiator should be as follows:
    - i. CPU : Active over the entire simulation time
    - ii. Display Processors and Video Decoder: Active for 50% of the simulation time
    - iii. Graphics Processor: Active for 20% of the simulation time
    - iv. Other initiators: Active over the entire simulation time.

For example, if a simulation is run for 1000 cycles, and the display processor has a total bandwidth requirement of 50%, it should request 100% of its bandwidth in 500 cycles, and stay idle for the remaining 500 cycles.

5. OCP data width: 4 bytes to 16 bytes
6. OCP interface frequency: 266 MHz to 800 MHz
7. Number of DRAM channels: 1 – 8
8. Number of parts per DRAM: 1 to 4 “x16” parts, assuming 64Mbytes per part
9. Frequency of DRAM bus: 333 MHz (DDR-2 667), 533 MHz (DDR-2 1066), 800 MHz (DDR-2/3 1600)
10. DRAM burst-length: 4 or 8 DRAM words. If DDR-3 is used, the burstlength must be 8.
11. Number of outstanding transactions per thread: 64 bytes to 512 bytes. Note that, the minimum value depends on the burst-length of the transaction.

## 5. Performance Analysis

In this section, we describe the essential performance analysis measures to make sure that the interconnection architecture is meeting the performance requirements of the system. Performance analysis consists of measuring the bandwidth at each initiator thread, worst case and average latency of the CPU, and the area overhead to achieve the bandwidth and latency.

The bandwidth at each initiator thread should be measured over multiple windows of time. A typical window size is 10000 clock cycles. In addition, the root mean square (RMS) error between the requested bandwidth and the serviced bandwidth should also be reported. For example, if a simulation runs for 100,000 cycles, and the bandwidth is measured in windows of 10,000 cycles, the requested and serviced bandwidth should be reported for each of the ten 10,000 clock cycle windows, and the value of  $\sum(\text{Requested bandwidth}(i) - \text{Serviced bandwidth}(i))^2$  should be reported, where “i” spans from 0 to 9, each “i” denoting a time window. Finally, the number of transactions completed, and the completion time of the last transaction should be reported. Bandwidth wise, if the RMS error is low, it means that the system is able to keep up with the bandwidth requirements of the thread. On the other hand, if the RMS error is high, it means that either the system is not able to keep up with the bandwidth requirements of the thread, or it is only able to sustain the bandwidth with high jitter. The completion time of the last transaction also indicates how well the system is keeping up with the bandwidth requirement. In general, earlier the last transaction completes, the better the system is keeping up with the bandwidth requirement. For the CPU threads, the average and worst case latency should also be reported.

In addition to meeting the performance of the system, it is necessary to provide the designer with a measure of the cost of achieving this performance. The cost is mainly area and power cost, both of which can be estimated by the number of storage elements in the system. The total number of storage elements should not only take the storage elements within the interconnection network, but the storage elements in the initiators as well.

The total number of outstanding transactions in the initiator gives a measure of the amount of storage available in the initiator. For example, if an initiator can generate 128 bytes of outstanding transactions, it means that the initiator has a FIFO with a storage of 128 bytes, into which the transactions are temporarily stored. Therefore, we can associate each outstanding transaction as an entry in the fifo. One can estimate the total amount of storage required in the system by the following formula:

$$\text{Measure of storage gate count} = \text{for all fifos in the network, } \sum(\text{fifo depth} * \text{fifo width in bytes} + \\ \text{for all initiators, for all threads } \sum(\text{number of outstanding transactions per thread}))$$

When presenting the performance (bandwidth, area etc described above) results, the values of the parameters described in Section 4.1 should also be reported. In addition, the number of transactions from each initiator thread (tag) should also be reported. Finally, the activity period of each initiator should be reported (starting cycle of activity and ending cycle of activity).

## 6. Conclusion

In this document, we addressed the performance benchmarking problem of Network-on-Chip architectures. We motivated the need for a performance benchmarking infrastructure, which interconnection network designers can utilize to measure the goodness of their architecture. Our focus in this work has been on performance benchmarking for SoCs targeted at video applications such as HDTV and set-top box. We also presented methods to analyze the performance of an interconnection architecture, by taking the performance goals and gate count into account. Future work will focus on other verticals in the scope of interconnection architecture design, such as wireless, automotive and multi-processor architectures.

## ACKNOWLEDGEMENTS

The authors would like to acknowledge the contribution of the other members of the Network-on-Chip Benchmarking Workgroup at OCP-IP.

## 7. References

- [1] W. Wolf, A. Jerraya, and G. Martin, “Multiprocessor System-on-Chip (MPSoC) Technology”, IEEE Transactions on Computer-Aided-Design of Integrated Circuits and Systems, Vol 27, Issue 10, Oct 2008, pp. 1701-1713
- [2] C. Grecu, A. Ivanov, A. Jantsch, P.P. Pande, E. Salminen, U. Ogras, and R. Marculescu, “Towards Open Network-on-Chip Benchmarks”, First International Symposium on Networks-on-Chip (NOCS’07), May 2007, pp. 205-205
- [3] A.L. Sangiovanni-Vincentelli, and Quo Vadis, “SLD:Reasoning about Trends and Challenges of System-level Design”, Proceedings of the IEEE, Mar. 2007, Vol. 95, Issue 3, pp. 467-506

- [4] Rikard Thid, Ingo Sander, Axel Jantsch, “Flexible Bus and NoC Performance Analysis with Configurable Synthetic Workloads”, EuroMicro Conference on Digital Systems Design, Aug 2006, pp. 681-688
- [5] Drew Wingard, “Achieving High Memory Bandwidth for Cortex Family based Video SoCs Using Multiple DRAM Channels”, ARM Developer’s Conference, 2008
- [6] Radu Marculescu, Umit Ogras, Li-Shiuan Peh, Natalie Enright Jerger, and Yatin Hoskote, “Outstanding Research Problems in NoC Design: System, MicroArchitecture, and Circuits Perspectives”, IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems (IEEE TCAD), Vol 28, No 1, January 2009