A Concurrent Architectural Model for Communication Sub-System

Ankur Agarwal, Fabiano Gomez Kovalski, Cyril Iskander, Ravi Shankar
Center for System Integration
Florida Atlantic University
Boca Raton, FL 33431

Abstract

Multiprocessor architectures and platforms such as multiprocessor system on chip (MPSoC) recently introduced to extend the applicability of the Moore's law, depend upon concurrency and synchronization in both software and hardware to enhance design productivity and system performance. These platforms will have to incorporate highly scalable, reusable, predictable, cost and energy efficient architectures. With the rapidly approaching billion transistors era, some of the main problem in deep sub-micron technologies characterized by gate lengths in the range of 60-90 nm will arise from non scalable wire delays, errors in signal integrity and un-synchronized communication. These problems may be addressed by the use of Network on Chip (NOC) (A Communication Sub-System) architecture for future SoC. NOC separates computing and communication concerns in an elegant manner. Such a platform can separate domain specific issues in separate layers, which will allow for more effective modeling of concurrency and synchronization issues, in an attempt to develop an optimized system. Such concurrency issues, if not addressed, may be detrimental to normal functioning of the system and may lead the system to a deadlock/livelock state. For such a layered architecture, models of computation (MOC) will provide a framework to model various algorithms and activities, while accounting for and exploiting concurrency and synchronization aspects. We have modeled a concurrent architecture for parametriziable, customizable, reusable and scalable NOC by integrating various models of computation. Varying network load under various traffic scenarios has been tested to extract the performance metrics. We show the simulation results for the performance parameters such as, power consumption, silicon area and latency.

Motorola Perspective

Network-on-Chip communication backbone is a quality-of-service (QoS) driven component based communication sub-system, which features customizable and concurrency compliant components. This communication sub-system is likely to enhance the productivity of system design phase as it provides an architectural platform for managing complexity and incorporate a reusable communication backbone. We propose to design a system as a layered architecture. The NOC design is based on layered architecture having two layers: network protocol layer and communication backbone layer. Other system layers such as architecture layer, RTOS layer and application layer will be integrated on this platform. One of the aims of this NOC implementation is to develop IP libraries for packet-switched communication backbones for multi-core systems. This reduces development time spent on the communication backbone for multi-core system, the cost of designing the final product and enhances productivity of system design. The developed libraries have reusable, customizable and parameterizable components, which are designed using MLDesigner tool. This software is for designing and analyzing systems on a multi-domain environment. The simulations on MLDesigner provide performance parameters, such as throughput, latency, resource usage, and network load. These parameters can be added with performance parameters extracted from hardware implementation such as, power consumption, maximum frequency of operation and cost (in terms of silicon area) to generate a specification set for different configuration of the NOC. This specification set can be used by the system architect to select the NOC configuration. Thus, system performance evaluations and what-if scenarios can be performed in the beginning of the design phase. This meets the requirements of the system to be designed without the need to invest time in designing the communication backbone from scratch.

1. Introduction

System complexity, driven by both increasing transistor count and customer need for increasingly savvy applications, has increased so dramatically that system design and integration can no longer be an after-thought. As a consequence, system level design, long the domain of a few expert senior engineers, is coming into its own as a discipline. For the field to grow and enhance engineering design productivity, many key concepts have had to be adopted from other domains.

Advances in semiconductor technologies have led to continual and rapid transistor scaling; this will soon facilitate the integration of a billion transistors on a small size integrated circuit. Product development time will increase from the current two years to five years or more, thanks to the increasing hardware complexity and the concomitant increasing sophistication of the applications demanded by the consumer.
Traditional SoC systems may integrate one or two general processors, with a few slaves as application specific processors (ASPs). The system is divided into several independent functional blocks, which are then designed, most likely as dedicated hardware engines (ASPs) or as tasks on one or two processors. These functional blocks synchronize their activities by means of local operating system on the processor(s) and an on-chip communication bus such as AMBA, to form a functional system [1]. With increasing user demands for computationally extensive applications on an embedded system, such as, multimedia, real-time video communication, and 3-D video gaming, this model is no longer feasible. Current Designs cannot be separated so clearly into hardware engines and software processes. QoS (Quality of Service) requirements for real-time performance, mobility, and flexibility (to address various consumer preferences) are too complex to be addressed so simplistically [2].

Moore's law predicts that a chip in 2010 will have more than four billion transistors operating in the multi GHz range [4], [7]. This is mainly due to the near-exponential decrease in the transistor size enabling faster transistor switching times and more densely integrated circuits. Such computation power has posed some challenges which include the disparity in transistor and wire speeds, and increased power dissipation, leading to a decrease in the area of the chip which can be utilized with a single clock cycle [8]. System designers also need to keep the power consumption of the system at a manageable level. Under such considerations, a single-processor implementation will not suffice. This has resulted in exploitation of multi-core architectures [2], thus driving the development of increasingly complex multi-processor-system-on-chips (MPSoCs) [2]. Such an MPSoC platform has set a new innovative trend for the real-time system designers. The consequences of this trend imply a shift in concern from computation and sequential algorithms to addressing concurrency, synchronization and communication issues in every aspect of hardware and software co-design and development. This has led to introduction of the multiprocessor-system-on-chip (MPSoC) platform [3]. As the technology scaling works better for transistors than for interconnecting wires [4], there is an increasing disparity between the wires and the transistors in terms of power consumption and latency. Thus, bus based communication becomes a bottleneck [5]. It is estimated that in 50 nm technology, global wire delays will reach up to 6-10 clock cycles [4], [8]. On a billion transistor chip, it would not be possible to send a global signal across the chip in real-time. As a result, achieving synchronization onto the system will be very difficult, if not unfeasible. Moreover a bus based SoC or MPSoC does not offer the required amount of reuse in order to meet the time-to-market requirements. Technology scaling has also had some unwanted side effects. This include cross coupling, noise, and transient errors [9]. This will continue to impact the productivity of system architects and designers [6].

NOC's can improve design productivity by supporting modularity and reuse of complex cores, thus, enabling a higher level of abstraction in the architectural modeling of future systems [8], [10], [11]. No delays are experienced for accessing the communication infrastructure, since multiple outstanding transactions originated by multiple cores can be handled at the same time, resulting in more efficient network resource utilization. However, given a certain network dimension (e.g., number of instantiated switches), large latency fluctuations for packet delivery could be experienced as a consequence of network congestion. Such delays would be unacceptable in real-time systems. There are two plausible solutions to this problem: (1) Over-dimensioning the network to achieve the worst case scenarios for a definite traffic pattern; (2) Providing priority levels to the traffic. The second solution is a cost-effective solution in terms of the power consumption. In this way, real-time requirements can be met by providing priority to real-time

traffic. Such real-time traffic can be guaranteed for its delivery to its destination by either reserving some paths for real-time data, or implementing priority-based scheduling criteria.

The key challenge for the system level design with integration of thousands of cores is the modeling of concurrency and synchronization issues. It requires representation of various activities and algorithms with appropriate Models of Computation (MOC). This defines our goals for this paper: (1) Model a packet based communication backbone in order to overcome the bus based limitations. (2) Such a backbone must be able to integrate several cores in order to provide a multi-core infrastructure, also know as NOC. (3) Design of packet based infrastructure must be reusable in order to overcome the limitations of conventional bus based systems. (4) Designed NOC must be a cost effective solution, so that it can be also be used for embedded systems. (5) Model the system functionality well in advance of building the actual computing system in order to provide a level of flexibility in system design. (6) Manipulate an abstract set of design elements simultaneously to generate different sets of QoS parameters and performance metrics and fine tune the system model.
23

2. NOC Background

From the communication perspective, there have been various topologies for the NOC architecture. Suggested network topologies include a star based NOC, which will communicate using the principles of CDMA [12]. Other researchers have proposed a tree based implementation of NOC, where each node in the tree behaves as a router in NOC [13]. In [14] researchers have compared various network topologies such as mesh, torus and ring among other for interconnection network in terms of latency, throughput, and energy dissipation. Several researchers have suggested a 2-D mesh architecture for NOC [15] as an efficient solution in terms of latency, power consumption and ease of implementation.

New algorithms have been proposed in this domain to reduce power consumption while securing cost optimization [16]. It has been a well established fact that such NOC architectures will be based on packet switched networks. This has led to new and efficient principles for design of routers for the NOC architecture [17]. These routers will be responsible for routing the entire traffic across and have to be interfaced with switches and resources in the NOC architecture. Various routing algorithms have been proposed for the NOC environment. Most of the researchers have suggested static routing algorithm and have performed communication analysis for NOC based on static behavior of the processes thus determining the static routing for NOC. They use communication dependency graph (CDG) to analyze inter process communication [18]. Other researchers provide a graph based application decomposition and mapping strategies for NOC [19], [20]. Researchers have also proposed RTOS architecture for NOC [21]. They provided some RTOS based application scheduling techniques on NOC. [22], [23], [24], [25], [15] [17] provides a graph based algorithms for application mapping onto NOC platform.

Based on the routing strategies, various design and implementation of router architectures have also been proposed. Wolkotte et. al. proposes circuit switched router architecture for NOC [26] while Dally and Towles have proposed router architecture for packet based switching [27]. Albenes and Susin provide, a wormhole based packet forwarding design and implementation of a NOC switch [29]. A. Z. Frederico, Santo, and Susin propose a fault tolerant routing protocol for NOC [29]. One of the main concerns in NOC is to be able to reduce the latency of operation. Therefore, there are various levels of the latency metrics that may be offered. Router architectures for supporting guaranteed bandwidth (GT) and best effort (BE) service have been proposed [30]. Design of a virtual channel is another important aspect of NOC. Virtual channel splits a single channel into two channels virtually providing two paths for the packets to be routed. Bjerregaard and Sparso have proposed the design and implementation of virtual channel router using asynchronous circuit techniques [31]. Another important component of NOC is the design of interconnects. Brager et. al. proposes transmission line based design of interconnect for NOC [32]. Morgenshtein et. al. propose a comparative analysis for serial vs parallel links for interconnect implementation [33].

The resource network interface (RNI) should be highly scalable and re-usable in order to be integrated with resources with different types of interfaces and data requirements. Substantial research has been conducted to propose the right data formats needed for various layers in the protocol stack. A reusable switch is used for effectively routing the packet through the entire NOC; they buffer packets at both input and output [34]. As buffer will occupy most of the area of NOC, thus various researches have studied buffer implementation.

Zimmer et. al. and Bolotin et. el. propose a simple implementation of buffer architecture for NOC [35], [36]. Zimmer et. al. have further implemented buffers using 0.18 µm technology to estimate the cost and area of buffers needed for NOC [37].
Need for implementation of fault tolerant, error detection, and error correction techniques is not certain for on chip implementation. QNOC implementation of NOC [38] argues that the communication strategies for on chip network may be considered reliable [40] while [41] proposes fault tolerant routing algorithm. Bertozzi, Binini, Micheli and Zimmer, Jantasch proposed the error detection and correction schemes for data on NOC link [42], [43]. Most of proposed architectures of NOC are yet to be implemented. Arteris is the only commercially available NOC implementation [44]. They have developed a NOC complier for targeting applications onto their NOC chip. Other existing implementations are QNOC and XPIPES. NOC will not be a useful concept until we are able to demonstrate an application performing better on an NOC as compared to bus based communication infrastructure. Several researchers have studied this issue. For example, Low density parity check (LDPC) decoder was implemented on a NOC platform [46]. More than 60% of hardware area of LDPC comprises of memory. NOC architecture, thus, was a suitable platform for this application. Jiang, Wolf and Chanradhar implemented a video application on NOC [46]. There is a need for NOC emulation platform in order to study the impact of various NOC topologies applications. Genko et. al. provide a FPGA based emulation platform for NOC [47].

Once the design of the basic NOC architecture became established, new techniques evolved to address advanced issues such as dynamic load balancing on to a node of the NOC architecture, the shortest/fastest path for the data flow through NOC, and energy efficient NOC architecture design. Most researchers have focused on the communication architecture of NOC; relatively few have focused on exploiting the computing capability of NOC. Lee et. al. have proposed an integrated NOC implementation [48].

3. NOC Issues and Challenges

In order to enhance system productivity, it is very important that an architect is able to abstract, represent and address most of the design issues and concerns at a high level of abstraction. There are many challenges to realizing this. System level design affords one the opportunity to review several different software-hardware architectures that meet the functional specifications equally well, to quickly trade-off among different QoS (Quality of Service) metrics such as latency, power, cost, size and ease of integration. Similarly, there are several issues related to NOC such as, the nature of the NOC Link, Link Length, Serial Vs Parallel Links, Bus vs Packet-based switching, and Leakage currents. In this section we discuss these issues.

3.1. Serial Vs Parallel Links

Transportation of data packets among various cores in a NOC can be performed by the use of either a serial or a parallel link. Parallel links make use of a buffer-based architecture and can be operated at a relatively lower clock rate in order to reduce power dissipation. However, these parallel links will incur high silicon cost due to inter-wire spacing, shielding and repeaters. This can be minimized up to certain limit by employing multiple metal layers. On the other hand, serial links allow saving in wire area, reduction in signal interference and noise, and further eliminate the need for having buffers. However, serial links would need serializer and de-serializer circuits to convert the data into the right format to be transported over the link and back to the cores. Serial links offer the advantages of simpler layout and simpler timing verification. Serial links sometimes suffer from ISI (Inter-symbol Interference) between successive signals while operating at high clock rates. But such drawbacks can be addressed by encoding and with asynchronous communication protocols.

3.2. Interconnect Optimization

Communication in a NOC is based on modules connected via a network of routers with links between the routers that comprise of long interconnects. Thus it is very important to optimize interconnects in order to achieve the required system performance. Timing optimization of global wires is typically performed by repeater insertion. Repeaters result in a significant increase in cost, area, and power consumption. Recent studies indicate that in the near future a large portion of chip resources will be used by inverters operating as repeaters. Thus there is a need for optimizing power on the NOC. Techniques for reducing dynamic power consumption include approaches discussed in [49], [50], [51]. Encoding is another effective way of

reducing dynamic power consumption [52]. In order to make NOC architectures more effective, innovative ways will have to be introduced to minimize the power consumed by the on-chip repeaters.

3.3. Leakage Power Consumption

Leakage current, which was negligible relative to dynamic switching current at larger transistor sizes (of 1 micron or more), is expected to dominate the current drain at sub-100 nm technologies. In a NOC, the link utilization rates vary and in many cases are very low, reaching a few percentage points. Networks are designed to operate at low link utilization in order to meet worst case scenario requirements, and thus having a higher link capacity helps reduce packet collisions. However, even when NOC links are idle they still will consume power in repeaters, due to the dominance of this leakage current at small feature sizes. Thus new techniques will have to evolve which will help reduce the leakage power consumption to make NOC architecture more effective.

3.4. Router Design

For embedded systems such as handheld devices, cost is a major driving force for the success of the product and therefore the underlying architecture as well. Along with being cost effective, handheld systems are required to be of small size and to consume significantly less power, relative to desktop systems. Under such considerations, there is a clear tradeoff in the design of a routing protocol. A complex routing protocol would further complicate the design of the router. This will consume more power and area without being cost effective. At the same time a simpler routing protocol will outperform in terms of cost and power consumption, but will be less effective in routing traffic across the system.

3.5. Quality of Service Challenges

Qualities of Service parameters for a NOC include latency, cost, power consumption, and silicon overhead in terms of the area that is required to support packet switched network. It is expected that various applications such as Real-Time, Multi-Media and Control, and computation-intensive algorithms such as, video encoding and decoding algorithm, 3-D Gaming, and Speech Recognition, will be supported on a NOC environment. Under such a scenario, NOC should be able to provide various levels of support for these applications.

3.6. Latency

NOC infrastructure must be able to guarantee a timely exchange of data packets for a real-time application. However, given a certain network size, large latency fluctuations for packet delivery could be experienced as a consequence of network congestion. Such variability and non-determinacy are obviously not acceptable for real-time applications. One possible solution to this problem is over-dimensioning the network by adding some redundant paths (links), nodes and buffers. These paths can be utilized when the main network is congested. Other possible solution to this problem would be to reserve some paths for real-time application in order to guarantee a timely delivery of data packets from one node to another node. Both the solutions are able to overcome the latency problem but increase the power consumption and the cost of NOC. A cost effective solution would to provide priority levels to the data traffic. Real-time traffic can be guaranteed its on-time delivery to its destination by either reserving some paths for real-time data, or implementing priority-based scheduling criteria. In such systems, we would be able to route the data packet for a real-time application in time but a data packet belonging to a lower priority application may be starved, without appropriate scheduling algorithms.

3.7. Cost, Power Consumption and Area Overhead

NOC infrastructure includes components responsible for packetization, transmission, and de-packetization of data. These components, respectively, are the network interface (NI), the virtual channel (VC) router, and the links. These components are repeated for every grid element in NOC. So if we consider a NOC with 3×3 mesh network, then it will have nine sets of components of NI, VC router and links. It can be clearly seen that these components will occupy a significant amount of silicon space on the chip and therefore the cost and the power consumption of the chip would increase. However it must be noted that serial packet based communication will still remain an optimum solution as compared to bus based system in terms of the power consumption and will reduce the cost of system design in the longer run due to the potential for reuse.

3.8. Consideration for a System Level Simulation Environment

A simulation environment must allow us to integrate hundreds of cores and model concurrency & synchronization issues. This requires representation of various activities and algorithms with appropriate MOCs. We define here the five main issues that should be met for a simulation environment to be suitable for system level designs. The simulation model must allow a system architect to: (1) Model the system functionality well in advance of building the actual computing system; (2) Model concurrency issues among various components in the system model; (3) Manipulate an abstract set of design elements simultaneously to generate different sets of QoS parameters and performance metrics, and to fine-tune the system model; (4) Integrate different MOCs onto this modeling environment; and (5) Access a well defined library of components defined in various MOCs so that the modeling time can be substantially reduced.

4. Communication Sub-System Implementation

We identified the components for Communication Sub-System from literature survey. We modeled all the components and their sub-components in a system level modeling environment-MLDesigner. We failed to integrate the designed components/sub-components. The main reasons for our failure were the process deadlocks, livelocks and resource starvations. Thus, we included concurrency modeling in our design flow. After concurrency analysis we debugged our modeling issues and integrated the sub-system.

Concurrency Model

Concurrency modeling step has been discussed in detail in a whitepaper "*Concurrency Model for Network-On-Chip Design Architecture"*. Please refer that paper for detailed explanation.

MLD Model

We have designed all the components in MLDesigner: A system level design and modeling environment. These components have been identified from high level specification in concurrency modeling. These components are identified as producer, consumer, buffer, scheduler and node. In this section, we discuss the specific implementation of NOC components.

4.2.1. Producer: Producer comprises of a resource and a protocol layer (network interface). The functions of producer are to generate a required traffic pattern and packetize the data in flits. Flit is the smallest unit of communication supported in designed NOC. The parameters for producer are: (1) The distribution pattern of the data; (2) The amount of data generated as a function of time (throughput); (3) Priorities of generated data - high, mid or low (QoS); (4) The source and the destination addresses for the data (route information). We are validating the designed NOC with various statistically generated traffic distribution patterns - uniform, exponential and Bernoulli. Producer outputs a flit when it receives a high-logic on "*buffAvail*" signal from the buffer. A flit is time stamped at the time of its generation. The time stamp field in the flit facilitates to monitor the latency involved in delivering the flit. The source and destination address field of the flit head, as shown in Figure 1, is updated at this time. It can be seen from the figure A that the first field of the flit header consist of the priority followed by the time stamp, x-direction of source address, y-direction of the source address, x-direction of the destination address and finally the y-direction of the destination address. The priority of this flit is governed as per the statistical distribution block. For example, in case of a uniform distribution pattern, every third flit will be a high priority flit. Once the new flit has its time stamp, source and destination address and priority fields updated, it is then forwarded to the output through "*dataOut*" signal. Figure 2 shows the MLD implementation of Producer.

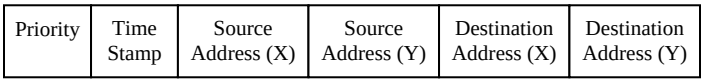

Figure 1: Flit Header

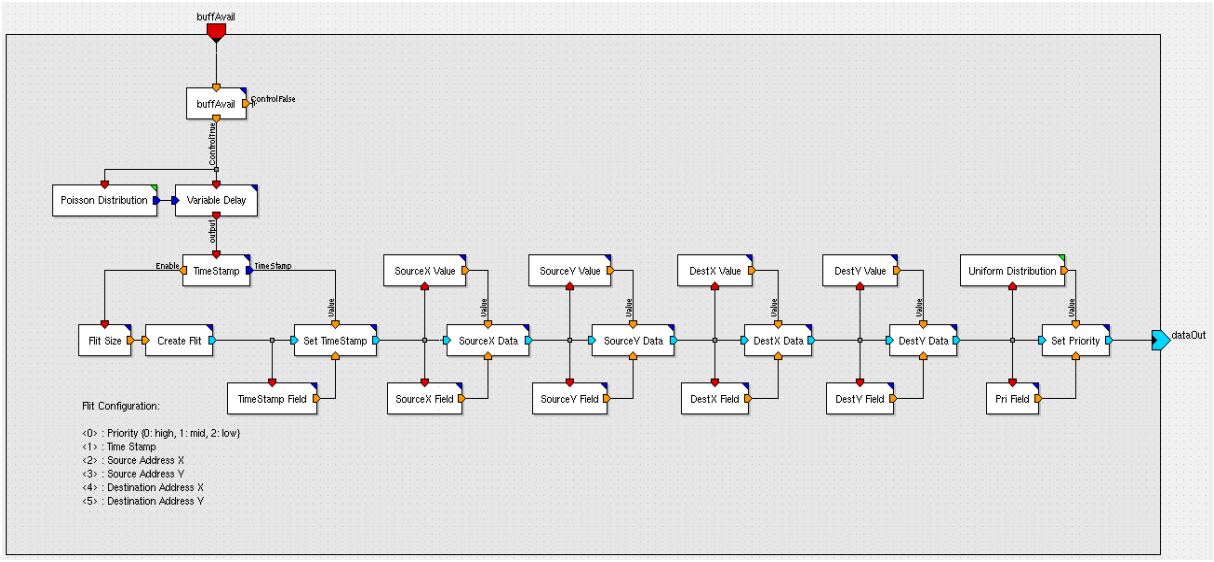

Figure 2: MLDesigner Implementation of the Producer

4.2.2. Input Buffer: The input buffer is responsible for storing the incoming flits, generating the proper handshaking signals to communicate with the scheduler and forwarding the flits to the next nodes. Data forwarding path has been implemented based on "request-grant" signaling approach. Incoming flits corresponding to all the priority levels (high, mid and low) are stored in a common buffer with "*buffSize"* as available space. High-logic and low-logic levels on "*buffAvail*" signals indicate whether the buffer is available or the buffer is full respectively. These data flits are then forwarded to the node based on their priority. High priority flits are forwarded before the mid or low priority flits. The availability of data flits in buffer for further transmission is indicated by "*shortDataInBuff*" and "*dataInBuff*" signals. The "*shortDataInBuff*" signals a high priority flit stored in the input buffer. Similarly, mid and low priority data flit stored in the buffer is indicated by "*dataInBuff*".

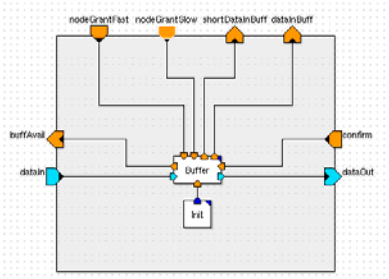

Figure 3: MLDesigner Implementation of the Buffer

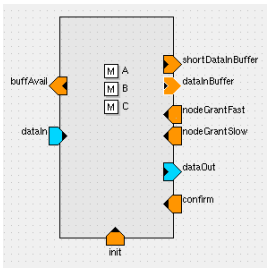

Figure 4: Internal Details of Input Buffer

If a grant comes in response any of these requests ("*nodeGrantFast"* for "*shortDataInBuff"* or "*nodeGrantSlow"* for "*dataInBuff"*), the flit stored in the buffer is forwarded to the corresponding Node. A Flit is not removed from the buffer until a confirmation (a high-logic on "*confirm"* signal) is received from scheduler process. The unavailability of output path for the data packet is signaled by a low-logic on "*confirm"* signal. In this case the data flit is not removed from the buffer and it is requested for being forward at a later time. Figure 3 shows the MLD implementation of buffer. Figure 4 shows the internal details of the model shown in figure 3. Initial condition in the buffer is generated by "*init*" signal. It depicts the amount of memory available at the start time of the simulation.

4.2.3. Scheduler: The scheduler handles the control part of the node process block. It is used for scheduling the incoming requests for data transmission to the next node, checking the availability of the output data path and arbitrating the requests from various input buffer associated with it. The data and control part have been separated (node handles the data part and scheduler handles the control signals) in order to manage

concurrency issues and make the design more scalable and reusable. Figure 5 shows the MLD implementation of scheduler and Figure 6 shows the internals of the scheduler. The model shown is a simplified model to better identify its connections.

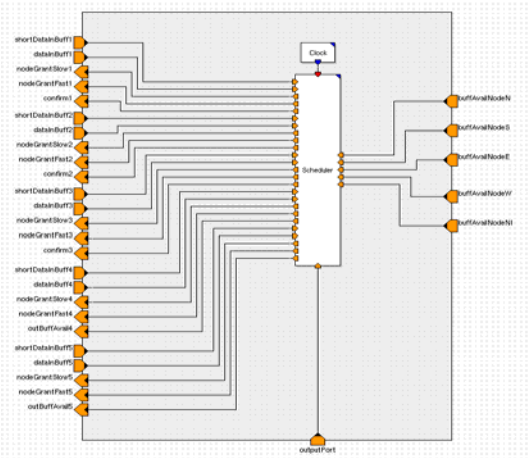

Figure 5: MLD Implementation of Scheduler

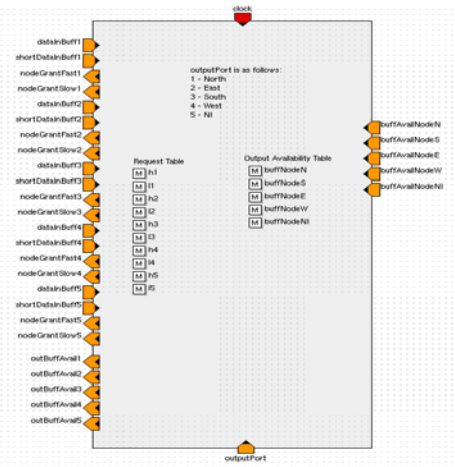

Figure 6: Internal Details of Scheduler

The scheduler is connected to five input buffers (one for each direction in 2-D mesh network and fifth buffer for connected with NI) and similarly, five buffers "*output buffers*" at the output side. The scheduler accepts the requests from input buffers at "*shortDataInBuff*" or "*dataInBuff*" signals and allocates the data path to these requests by asserting the grant signals "*nodeGrantFast*" or "*nodeGrantSlow*" respectively. The data path allocation is based on the availability of the output path and node. For handling multiple requests from the input buffers, priority-based-round-robin scheduling discipline has been implemented. Node is responsible for informing the scheduler the physical output path for flit transmission by "*outputPort*" signal. The scheduler then checks the availability of the output data path. In case of unavailability of output buffer, a low-logic on "*confirm*" signal is transmitted back to the respective input buffer while the availability of the data path is acknowledged by a high-logic on "*confirm*" signal.

For modeling such scheduler functionality in a system level environment, request and output path availability tables have been implemented inside scheduler. These tables are updated by their corresponding signals. This takes care of the credit flow control process in the network. In order to maintain priority based round robin discipline, a separate memory space has been allocated for individual signals into the request table. Thus, the scheduler first allocates the requests corresponding to high priority data request signals and then serves the mid and low priority data requests. The number of available credit for the output buffers are updated in output path availability table by corresponding output signals.

4.2.4. Node: Node is responsible for calculating the output path and handling the actual data transfer on the implemented backbone. Figure 7 shows the MLD implementation of Node and Figure 8 shows the internal connections of node. X-Y routing discipline has been implemented in Node for determining the output path. A turn-prohibition method has been employed for avoiding the deadlock/livelock situation in the network. Upon on the receipt of the data, node extracts the destination information and determines the physical output port for transmitting the data. This output port address is sent to the scheduler, which determines the availability of this port. Upon its availability, data flits are then forwarded to the corresponding output buffer in this port.

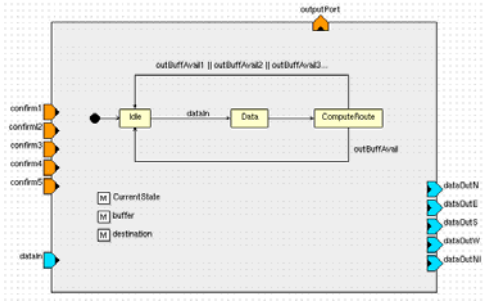

Figure 7: MLD Implementation of Node

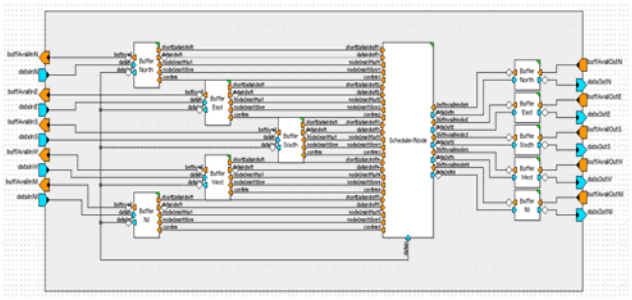

Figure 8: Internal Details of Node

4.2..5. Output Buffer: Output buffer accepts the incoming flits from the node and forwards these flits to the input buffer for the next node. It is implemented in form of two concurrently executing state machines. Figures 9 and 10 show the output buffer model and its internal details respectively. Upon the receipt of the data flits, these are placed in the buffer memory. The input state machine keeps accepting and storing the data flits while this buffer memory is available. Output buffer signals the buffer space availability by a high-logic on "*buffAvail*" signal and the unavailability of this memory pace is indicated by a low-logic on "*buffAvail*" signal. This status is informed to the scheduler for determining the available of the output data path. The output state machine senses the request for transmitting the data from the input buffer of the next node by "*outputBuffAvail*" signal and forwards the data flit if available in the buffer memory space.

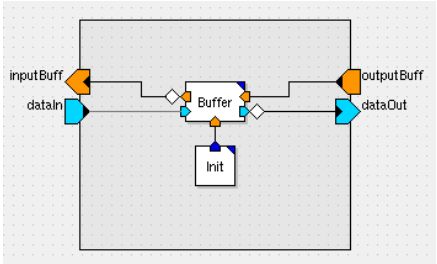

Figure 9: MLD Implementation of Output Buffer

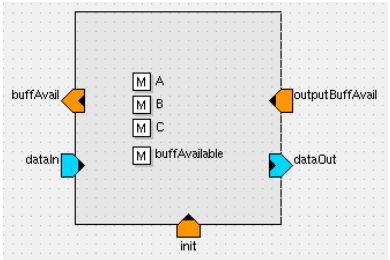

Figure 10: Internal Details of Output Buffer

4.2.6. Consumer: Consumer comprises of a resource and a protocol layer (network interface). The functions of consumer are to accept data flits, extract the data by striping the header information and to forward this data to the connected resource.

4.2.7. System Integration & Simulation: A 4×4 mesh network for NOC is shown in Figure 12. Choice of a right simulation environment is a very important step. We studied various simulation platforms such as, Ptolemy, MESH, OPNET and MLD. We find MLD suitable for our work. We have modeled our components using FSM, SDF (synchronous data flow) and DE domain. The top level simulation on the final model will be performed in DE domain, thus combining various MOCs as discussed earlier. DE domain simulations are based on events, thus making it an idle choice for running the final simulations faster.

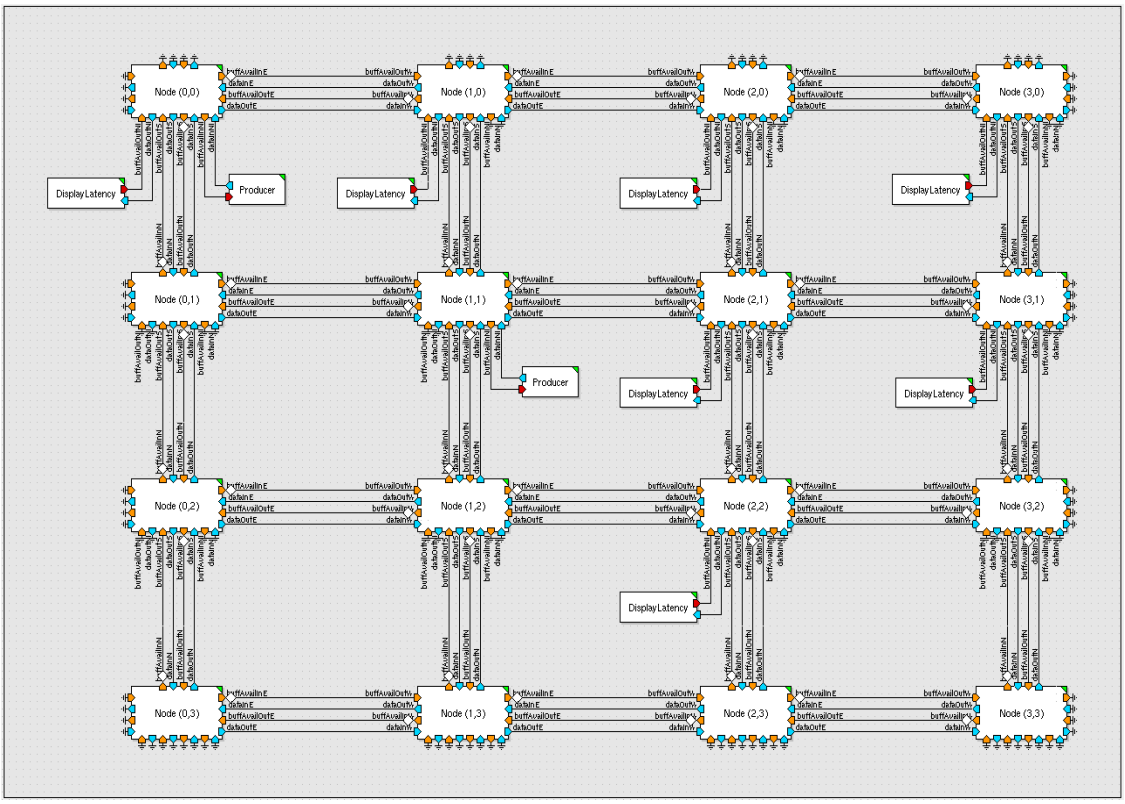

Figure 12: A 4×4 mesh network for Communication Sub-System

Performance Evaluations

For performance evaluation we propose to extract the performance parameters from FPGA based rapid prototyping of the processes in communication sub-system and literature survey. We are modeling all the sub-system processes on a prototyping environment. From this environment we will extract the performance parameters such as latency, power and area. Along with these, we will extract the power consumption and latency figures from the literature survey as well. The parameters will then be fed to MLD in form of performance library. With this knowledge a system designer will be able to performance architectural trade-off at a system level. We propose to use MLD environment for performance evaluation.

5. Simulation Results

Figure 14 shows the simulation results for the latency of hi-priority data for various buffer sizes. For simulation the latency, about 800 flits were injected into the network. The latency figures were plotted for the buffer size from 2 to 10. It can be seen from the figure that most of the hi-priority flits have the latency of about 10 clock cycles. This is due to the fact that about 6 clock cycles is being used in generating the flit, forwarding the flit to the input buffer, forwarding the flit from input buffer to node and consuming the flit at the node. Moreover in case of two or more hops, the network will experience a slight increase in the latency. Further it can be realized from the figure that with the smaller buffer size there is less latency than in case of the larger buffer size.

Similarly, in Figure 15 we show the latency of mid-priority data. The network is again injected with 800 flits. The network latency s plotted for various buffer sizes. The network suffers the average latency of about 16 clock cycle. The latency seems to be almost independent of buffer of size in this case.

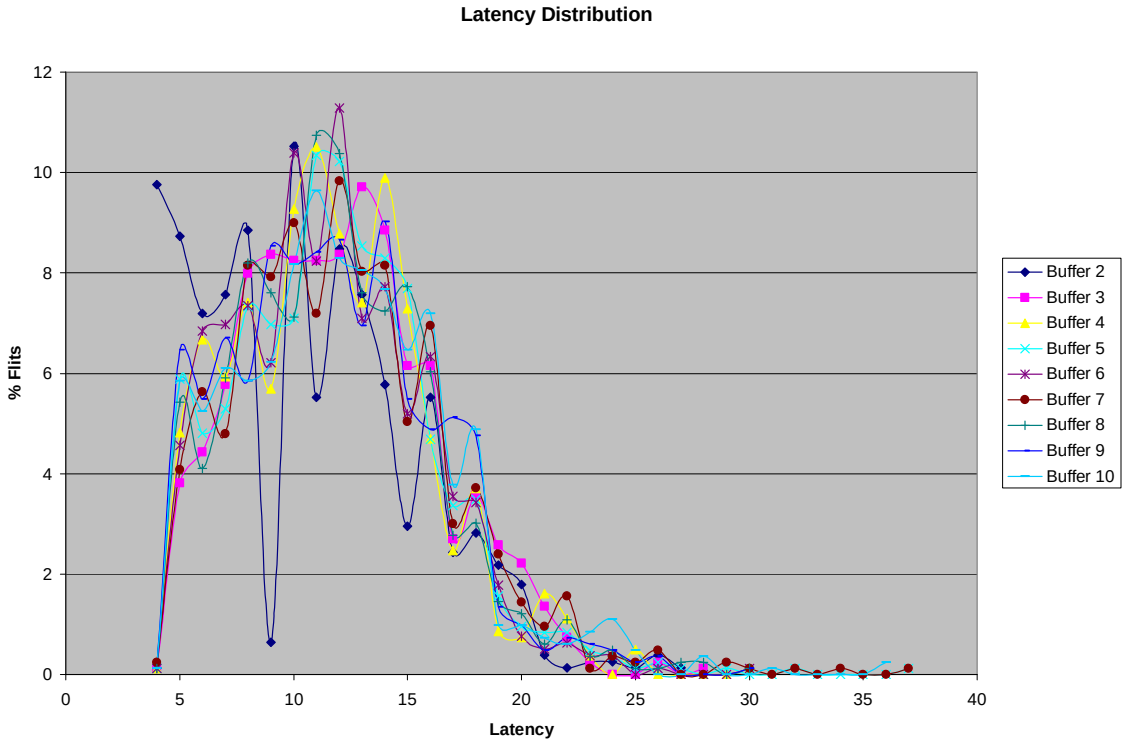

Figure 14: Latency of High Priority Data for Various Buffer Sizes

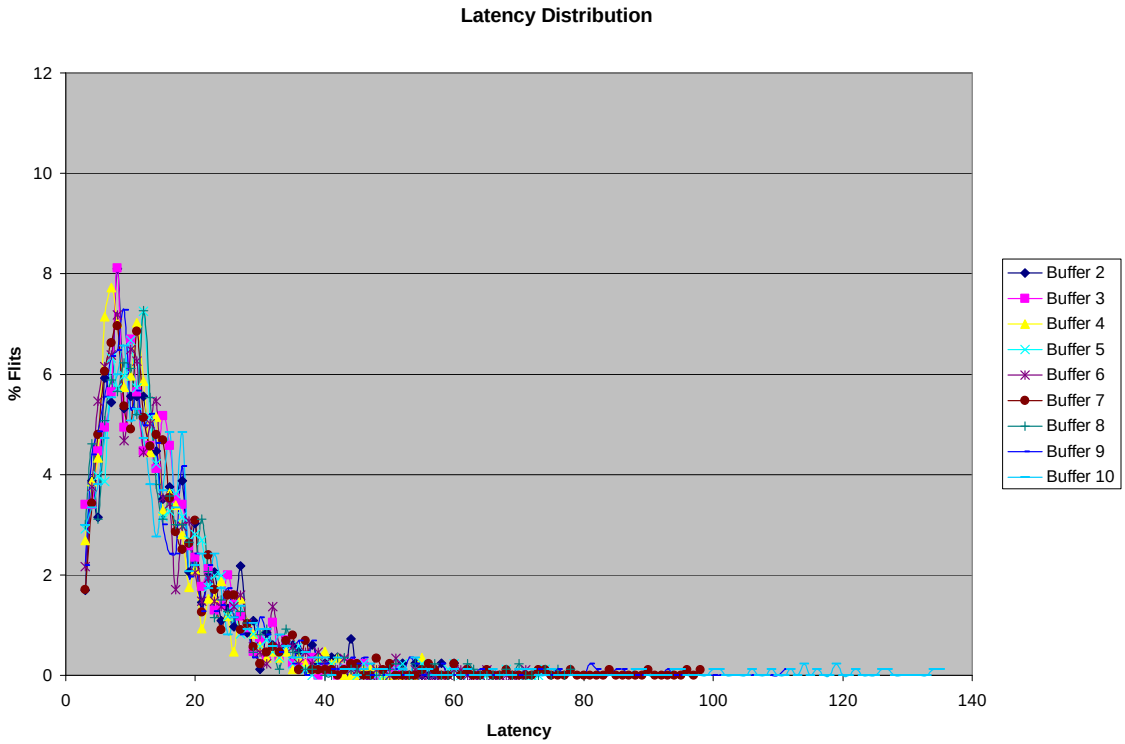

Figure 15: Latency of Mid Priority Data for Various Buffer Sizes

6. Conclusion

Communication sub-system will provide a reusable communication backbone for multicore architectures. The system designers will not have to worry about designing complex bus based system. Such a communication system further supports the multi-core architecture and provides less latency and consumes less power than based system. It an elegant architecture for managing complexity of design of a system by addressing concurrency and synchronization issues in the system.

References

[1] G. Desoli, E. Filippi, "An Outlook on the Evolution of Mobile Terminals: From Monolithic to modular multi-radio, multi-application platforms", IEEE magazine on Circuits and Systems, vol. 6, No. 2 2006, pp. 17-29

[2] K. Keutzer, S. Malik, A. Richard Newton, Jan M. Rabaey, A. Sangiovanni-Vincentelli, System level design: Orthogonalization of concerns and platform based design, *IEEE Transaction on CAD of Integrated Circuits and Systems*, 19(12): 2000, 1523-1543

[3] J. Ahmed Meine, Wolf Wayne, *MULTIPROCESSOR SYSTEM-ON-CHIPS*. Morgan Kaufamann Publisher, 2005.

[4] L. Benini and G. De Micheli. Networks on chip: a new SOC paradigm, *IEEE Computer*, vol. 35 No. 1, January 2002, 70-78

[5] A. Hemani, Axel Jantsch, Shashi Kumar Adam Postula, Johnny Öberg, Mikael Millberg, Dan Lindqvist, Network on Chip: an architecture for billion transistor era, *Proc. of IEEE NorChip Conference*, November 2000, 8-12

[6] D. Bertozzi and L. Benini, "Xpipes: A network-on-chip architecture for gigascale system-on-chip," *IEEE Circuits and Systems*., vol. 4, no.1 pp. 18-31, 2004

[7] X. Jiang, W. Wolf, J. Hankel, S. Charkdhar, "A methodology for design, modeling and analysis for networks-on-Chip," *IEEE International Symposium on Circuits and Systems*, pp. 1778-1781 May 2005

[8] A. Hemani, Axel Jantsch, Shashi Kumar Adam Postula, Johnny Öberg, Mikael Millberg, Dan Lindqvist, Network on Chip: an architecture for billion transistor era, *Proc. of IEEE NorChip Conference*, November 2000, 8-12

[9] S. Kumar, A. Jantsch, J-P. Soininen, M. Forsell, M. Millberg, J. Oberg, K. Tiensyrja, and A. Hemani, "A Network on Chip Architecture and Design Methodology". *In IEEE Computer Society Annual symposium on VLSI*, April 2002, 117-124

[10] A. M. Amory, Érika Cota, M. Lubaszewski, Fernando G. Moraes, Reducing test time with processor reuse in network-on-chip based systems**,** *Proceedings of the 17th ACM symposium on Integrated circuits and system design*, 2004, 111 - 116

[11] A. Jantsch and H. Tenhunen. *Networks on Chip* (Kluwer Academic Publisher, 2003)

[12] A. Agarwal, R. Shankar, "A Layered Architecture for NOC Design methodology", IASTED International Conference on parallel and Distributed Computing and Systems, 2005, pp. 659-666

[13] D. Kim, M. Kim, G.E. Sobelman, **"**CDMA-based network-on-chip architecture", IEEE Asia-Pasific Conference on Circuits and Systems, vol. 1, pp. 137-140, December 2004

[14] A. Adriahantenaina, H. Charlery, A. Greiner, L. Mortiez, C.A. Zeferino, C.A., "SPIN: a scalable, packet switched, on-chip micro-network, IEEE Conference and exhibition on, Design, Automation and Test in Europe Conference and Exhibition, pp. 70-73, 2003

[15] P. Pratim Pande, C. Grecu, M. Jones, A. Ivanov, R. Saleh, "Performance evaluation and design trade-offs for network-on-chip interconnect architectures", IEEE transactions on Computers, vol. 54, Issue 8, pp. 1025-1040, Aug. 2005

[16] L. Benini, D. Bertozzi, "Network on chip architecture and design methods", *IEEE proceeding Computation Digital Technology*, vol. 152, No. 2, March 2005, 261-271

[17] P. Bhojwani, R. Mahapatra, Jung Kim Eun, T. Chen, "A heuristic for peak power constrained design of network-on-chip (NoC) based multimode systems", IEEE International Conference on VLSI Design, pp. 124-129, 2005.

[18] D. Rostilav, V. Vishnyakov, E. Friedman, R. Ginosar, An asynchronous router for multiple service levels networks on chip, *11th IEEE international symposium on asynchronous circuits and systems*, March 2005, 44-53

[19] A. Siebenborn, O. Bringmann, W. Rosenstiel, "Communication analysis for network-on-chip design", IEEE International conference on Parallel Computing in Electrical Engineering, pp. 315-320, September 2004

[20] J. Hu, R. Marculescu, “Energy-aware Communication and task scheduling for network-on-chip architectures under real-time constraints”, IEEE Conference and Exhibition on Design, Automation and Test in Europe, vol. 1, pp. 234-239, Febrary 2004.
[21] U.Y. Ogras, R. Marculescu, “Energy- and performance-driven NoC communication architecture synthesis using a decomposition approach”, IEEE Conference and exhibition on Design, Automation and Test in Europe, vol. 1, pp. 352-357, 2005.
[22] J. Madsen, S. Mahadevan, K. Virk, M. Gonzalez, “Network-on-chip modeling for system-level multiprocessor simulation”, IEEE 14th Conference on Real-Time System, pp. 265-274, 2003.
[23] J. Hu., R. Marculescu, “Energy-aware Mapping for tile-based NoC architectures under performance constraints”, Asia and South Pacific IEEE Proceedings on Design Automation Conference, pp. 233-239, January 2003.
[24] L. Tang, S. Kumar, “Algorithms and tools for network on chip based system design”, 16th IEEE Proceeding on Integrated Circuits and Systems Design, pp. 163-168, September 2003.
[25] K. Srinivasan, K.S. Chatha, G. Konjevod, “Linear programming based techniques for synthesis of network-on-chip architectures”, IEEE International conference on Computer Design: VLSI in Computers and Processors, pp. 422-429, 2004.
[26] R. Chae-Eun, J. Han-You, Ha Soonhoi, “Many-to-many core-switch mapping in 2-D mesh NoC architectures”, IEEE International conference on Computer Design: VLSI in Computers and Processors, pp. 438-443, Oct. 2004.
[27] J. Hu, R. Marculescu, “Energy- and performance-aware mapping for regular NoC architectures”, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 4, Issue 4, pp. 551-562, April 2005
[28] S. Murali, L. Benini, G. de Micheli, “Mapping and physical planning of networks-on-chip architectures with quality-of-service guarantees”, Asia and South Pacific IEEE proceeding on Design Automation Conference, vol. 1, pp. 27-32, January 2005.
[29] P.T. Wolkotte, G.J.M. Smit, G.K. Rauwerda, L.T. Smit, “An Energy-Efficient Reconfigurable Circuit-Switched Network-on-Chip”, 19th IEEE International Conference on Parallel and Distributed Processing Symposium, pp. 155-163, 2005
[30] W. J. Dally, B. Towles, “Route Packets, Not Wires: On-Chip Interconnection Networks”, IEEE International Conference on Design and Automation, pp. 684-689, June 2001.
[31] C. Albenes Zeferino Frederico G. M. E. Santo, Altarniro Amadeu Susin, “ParlS: A Parameterizable Interconnect Switch for Networks-on-Chips”, ACM Conference, pp. 204-209, 2004.
[32] A. Pullini, Federico Angiolini, D. Bertozzi, L. Benini, “Fault Tolerance Overhead in Network-on-Chip Flow Control Schemes”, ACM Conference, pp. 224-229, 2005
[33] E. Rijpkema, K. Goossens, A. Radulescu, J. Dielissen, J. van Meerbergen, P. Wielage, E. Waterlander, “Trade-offs in the design of a router with both guaranteed and best-effort services for networks on chip”, IEEE Proceeding on Computers and Digital Techniques, vol. 150, Issue 5, 22, pp:294-302, September 2003
[34] T. Bjerregaard, J. Sparso, “Virtual channel designs for guaranteeing bandwidth in asynchronous network-on-chip”, IEEE Proceedings on Norchip Conference, pp. 269 – 272, November 2004
[35] A. Barger, D. Goren, A. Kolodny, “Design and modelling of network on chip interconnects using transmission lines”, 11th IEEE International Conference on Electronics, Circuits and Systems, pp. 403-406, December 2004.
[36] A. Morgenshtein, I. Cidon, A. Kolodny, R. Ginosar, “Comparative analysis of serial vs parallel links in NoC”, IEEE International Proceedings on System-on-Chip, pp. 185-188, November 2004
[37] Yi Ran Sun, S. Kumar, A. Jain, Simulation and evaluation for network on chip architecture using NS-2, 20th IEEE *International Conference preceding for NorChip* vol. 5, May 2003
[38] I. Saastamoinen, M. Alho, J. Nurmi, “Buffer implementation for Proteo network-on-chip”, International IEEE Proceesing on Circuits and Systems, vol. 2 pp. 113-116, May 2003
[39] H. Zimmer, S. Zink, T. Hollstein, M. Glesner, “Buffer-architecture exploration for routers in a hierarchical network-on-chip”, 19th IEEE International Proceeding on Parallel and Distributed Processing Symposium April 2005.
[40] E. Bolotin, A. Morgenshtein, I. Cidon, R. Ginosar, A. Kolodny, “Automatic Hardware-efficient SoC Integration by QoS Network-on-Chip”, 11th International IEEE Conference on Electronics, Circuits and Systems, pp. 479-482, December 2004

[41] M. Pirretti, G.M. Link, R.R. Brooks, N. Vijaykrishnan, M. Kandemir, M.J. Irwin, "Fault tolerant algorithms for network-on-chip interconnect", Annual IEEE Proceeding on Computer Society, pp. 46-51, February 2004
[42] D. Bertozzi, L. Benini, G. De Micheli, "Error control schemes for on-chip communication links: the energy-reliability tradeoff", IEEE Transaction on Computer-Aided Design of Integrated Circuits and Systems, vol. 24, Issue 6, pp. 818-831, June 2005
[43] H. Zimmer, A. Jantsch, "A fault model notation and error-control scheme for switch-to-switch buses in a network-on-chip", First International IEEE/ACM/IFIP conference on Hardware/Software Codesign and System Synthesis, pp. 188-193, October 2003
[44] Arteris: A Netwrok-on-Chip Company, www.arteris.com
[45] T. Theocharides, G. Link, N. Vijaykrishnan, M.J. Irwin, "Implementing LDPC decoding on network-on-chip", 18th IEEE International Conference on VLSI Design, pp. 134-137, January 2005
[46] X. Jiang, W. Wolf, J. Henkel, S. Chakradhar, T. Lv, "A case study in networks-on-chip design for embedded video", European IEEE Conference and Exhibition on Design, Automation and Test, vol. 2, pp.770-775, February 2004.
[47] N. Genko, D. Atienza, G. De Micheli, J.M. Mendias, R. Hermida, F. Catthoor, "A complete network-on-chip emulation framework", European IEEE Conference and Exhibition on Design, Automation and Test, vol. 1, pp.246-251, February 2005
[48] S. J. Lee, K. Lee, S.J. Song, H.J. Yoo, "Packet-switched on-chip interconnection network for system-on-chip applications", IEEE Transaction on Circuits and Systems II, vol. 52, Issue 6, pp. 308-312, June 2005
[49] A. Morgenshtein, I. Cidon, A. Kolodny, R. Ginosar, "Low-leakage repeaters for NoC interconnects", IEEE International Symposium on Circuits and systems, ISCAS 2005, vol. 1, pp. 600-603
[50] N. Chabini, W. Wolf, "Reducing dynamic power consumption in synchronous sequential digital designs using retiming and supply voltage scaling", IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 12, Issue 6, 2004 pp.573 – 589
[51] L. Yan, L. Jiong N.K. Jha, "Joint dynamic voltage scaling and adaptive body biasing for heterogeneous distributed real-time embedded systems", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 24, Issue 7, 2005 pp.1030 – 1041
[52] Le Yan; Jiong Luo; Jha, N.K.; "Combined dynamic voltage scaling and adaptive body biasing for heterogeneous distributed real-time embedded systems", IEEE International Conference on Computer Aided design ICCAD 2003, pp. 30-37
[53] Chun-Gi Lyuh; K. Taewhan, "Low power bus encoding with crosstalk delay elimination [SoC]", 15th Annual IEEE International Conference on ASIC/SOC, September 2002, pp. 389-393