A Network on Chip Architecture and Design Methodology

Shashi Kumar[1], Axel Jantsch[1], Juha-Pekka Soininen[2], Martti Forsell[2],
Mikael Millberg[1], Johny Öberg[1], Kari Tiensyrjä[2] and Ahmed Hemani[3]
[1] *Laboratory of Electronics and Computer Systems, Department of Microelectronics and Information Technology, Royal Institute of Technology, 164 40 Kista, Stockholm, Sweden*
[2] *VTT Electronics, Box 1100, Oulu, FIN-90571, Finland*
[3] *Spirea AB, Kista Science Park, Electrum 209, S-164 40, Stockholm*

Abstract

We propose a packet switched platform for single chip systems which scales well to an arbitrary number of processor like resources. The platform, which we call Network-on-Chip (NOC), includes both the architecture and the design methodology.

The NOC architecture is a $m \times n$ mesh of switches and resources are placed on the slots formed by the switches. We assume a direct layout of the 2-D mesh of switches and resources providing physical- architectural level design integration. Each switch is connected to one resource and four neighboring switches, and each resource is connected to one switch. A resource can be a processor core, memory, an FPGA, a custom hardware block or any other intellectual property (IP) block, which fits into the available slot and complies with the interface of the NOC. The NOC architecture essentially is the on-chip communication infrastructure comprising the physical layer, the data link layer and the network layer of the OSI protocol stack. We define the concept of a region, which occupies an area of any number of resources and switches. This concept allows the NOC to accommodate large resources such as large memory banks, FPGA areas, or special purpose computation resources such as high performance multi-processors.

The NOC design methodology consists of two phases. In the first phase a concrete architecture is derived from the general NOC template. The concrete architecture defines the number of switches and shape of the network, the kind and shape of regions and the number and kind of resources. The second phase maps the application onto the concrete architecture to form a concrete product.

1. Introduction

Current algorithm on chip and system on chip design methodologies cannot respond to the needs of the billion-transistor area. The design would take too much time and mapping of applications to dedicated architectures would be impossible. The possible solutions must be searched from platform based design and computer system design, which rely on the reuse of components, architectures, applications and implementations. The essential issue is the trade-off between generality and performance. Generality provides reusability of hardware, operating systems and development practices, while performance (delay, cost, power, etc.) is achieved by using application specific structures.

We propose a NOC platform, consisting of architecture and design methodology, which scales from a few dozens to several hundred or even thousands of resources. A resource may be a processor core, a DSP core, an FPGA block, a dedicated HW block, a mixed signal block, or a memory block of any kind such as RAM, ROM or CAM. We base this proposal on three assumptions:

1. Moore's law will continue to hold for another five to 15 years. In that case our platform should prove useful in the time period 2005-2015 [1].
2. Single processors will not be able to utilize the transistors of an entire chip. Single synchronous clock regions will span only a small fraction of the chip area [16, 2, 3].
3. Applications will be modeled as a large number of communicating tasks. The different tasks may have very different characteristics (e.g. control or data flow dominated) and origins (most of them are reused from earlier products or from external sources) [4]. This will make a heterogeneous implementation with different kind of resources for different tasks the most cost effective solution.

From this we conclude that a large number of different kinds of blocks, each of the size of a few hundred thousand gates, will constitute the computational resources. They have to be connected efficiently.

Increasing non-recurring cost of these chips require that design cost of chips must be shared across applications. Furthermore, the same or different variants of the same application have to be mapped onto different variants of

Proceedings of the IEEE Computer Society Annual Symposium on VLSI (ISVLSI'02)
0-7695-1486-3/02 $17.00 © 2002 IEEE

the product, each establishing a different solution of the cost/performance/functionality trade-off. If this can be done quickly and cost effectively, many product versions for various market niches can be supported. Physical level and architectural level design integration will be very useful for this. This implies that physical layout and implementation issues are kept in mind while taking architectural decisions, or the architectural design is carried out within constraints of physical size and a floor plan.

The proposed NOC platform would effectively separate the specification of inter-task communication from the implementation of that communication; separate the design, implementation and verification of individual tasks from the rest of the application (a precondition for task reuse); separate the development, optimization and verification of the individual resource from the network infrastructure. We argue that the consequent separation of different concerns is a way to develop high-performance, cost-effective products while boosting design productivity.

Here is not the place to speculate about the kind of products to be expected within five to ten years. However, we assume that the future devices will have the following requirements and features:

1. Processing of multiple ultra high data rate (> 100 MB/sec) streams of data including audio and video data. The devices will be required to store this data and process it in real time.
2. Devices will be multi-functional. The functionality could be a mix of entertainment (like games, music instruments), communication, remote control, surveillance etc.
3. Devices will have high-capacity wire line or more likely wireless interfaces to standard networks like telephone network, Internet, and will need to be able to handle multiple communication protocols simultaneously
4. Security and secrecy of data stored and flowing through these devices will become important.

Clearly, a NOC based design will not always be the preferred solution for all kinds of applications. We expect that NOC based designs will provide good solutions for flexible products that should be reconfigurable and programmable; for designs which are the basis for several product variants; for applications with a heterogeneous task mix; for applications with stringent time to market requirements; for products where reuse both at the block and the function and feature level is considered valuable.

The design costs can be justified by increasing the implementation volumes and it is likely that the billion-transistor chips are not designed for single product instances or single applications. The design methodology must therefore support product family management. Tolerance of incomplete specifications, management of configurations and modifications, support for multiple languages and methods, and capability to handle different abstraction levels simultaneously are desirable characteristics.

Verification and testing are ever increasing challenges in today's design routines. With every new technology generation they are becoming more pressing. We argue that the NOC platform effectively addresses these challenges by separating the computation resources from each other and from the communication network for all issues of design, verification and testing.

In section 2, we list some other research work related to complex system design on a chip. In section 3 we describe the basic ideas and concepts of our proposed NOC architecture. In section 4 we describe the principles of design methodology for NOC based systems. In section 5 we discuss issues of physical implementation and performance for NOC architecture.

2. Related work

It is being realized, by all research groups involved in system level design, that it is absolutely necessary to allow reuse of already designed components or blocks. Gajski et. al. [5] have proposed an IP-centric embedded system design methodology. The major challenges in the IP centric methodologies are the interface synthesis among various IP blocks and system verification. Recently, Platform Based Design methodology [6] has been proposed which not only allows reuse of components but also reuse of system architectures and topologies. The basic idea is that an architecture, which is suitable and efficient for one application will also be suitable and efficient for many similar applications. The idea of using the same architecture (platform) for development of application not only speeds up application design but also reduces its verification time. Keutzer et. al. [7] have extended the idea of platform based design by including a layer of software on top of the hardware platform to help application development. This layer is called Software Platform. The combination of hardware and software platforms is referred as System Platform. It has also been realized that the key to reuse and integration of IP components is the communication from the physical to the system and conceptual level, and consequently communication centric architectures, platforms and methodologies have been developed [8, 9, 10].

Many architectural templates have been proposed for hardware platforms for future SoCs. There is a general emphasis on providing efficient and standardized communication infrastructure for connecting multiple resources on the chip [11, 8, 9]. There is a trend to adapt layered approach of OSI reference model towards on Chip communication [12, 10, 13].

It is estimated that video and audio processing are going to be common tasks in many applications. These applications are going to require storage and processing of large amount of data. It is predicted that memories are going to take around two third of the chip area in future system on chips [14]. Many researchers have concentrated on analyzing hierarchical organizations of memories and

optimization of memory sizes and data storage strategies for data intensive applications [15]. Researchers have alos simulated theoretically elegant shared memory model on message passing parallel computers in order to develop data intensive applications on them [19].

The future system on a chip, incorporating many different types of processing and memory elements, has to operate using Globally Asynchronous Locally Synchronous (GALS) paradigm [16], at least at the hardware level. GALS paradigm not only avoids the problem of clock skew but also leads to lower power consumption.

3. Network on Chip Architecture

The NOC architecture provides the communication infrastructure for the resources. We have two main objectives. Firstly, it is possible to develop the hardware of resources independently as stand-alone blocks and create the NOC by connecting the blocks as elements in the network. Secondly, the scalable and configurable network is a flexible platform that can be adapted to the needs of different workloads, while maintaining the generality of application development methods and practices.

3.1. The NOC network

We chose a simple mesh interconnection topology as basic topology, because it is simplest from a layout perspective and the local interconnections between resources and switches are independent of the size of the network. Moreover, routing in a two-dimensional mesh is easy resulting in potentially small switches, high capacity, short clock cycle, and overall scalability.

A NOC consists (Figure 1) of resources and switches that are connected using channels as a mesh (Manhattan-like structure) so that they are able to communicate with each other by sending messages. A resource *R* is a computation or storage unit or their combination. A switch *S* (Figure 2) routes and buffers messages between resources. Each switch is connected to four other neighboring switches through input and output channels. A channel *C* consists of two one-directional point-to-point buses between two switches or a resource and a switch. Switches may have internal queues to handle congestion. We call this approach *Chip-Level Integration of Communicating Heterogeneous Elements* (CLICHÉ).

The precise layout and geometry depends on the technology generation. We expect that the area of a resource is the maximal synchronous region in a given technology. It is expected to shrink with every new technology generation. Consequently the number of resources will grow, the switch-to-switch and the switch-to-resource bandwidth will grow, but the network wide communication protocols will be unaffected. Figure 1 illustrates the principles of the physical floor plan within the NOC. Consider a 60nm CMOS technology expected in 2008, a 22mm × 22mm chip size, and a resource size of 2mm × 2mm and a minimum wire pitch of 300nm. A NOC would accommodate 10 × 10 resources, each switch would occupy 30μm × 30μm and the channels would be 30μm wide. Assuming that we can use 3 metal layers for the switch-to-switch connection we have space for 300 wires. Since we need control, handshaking and signaling bits will yield an effective data bus width of 256 bits.

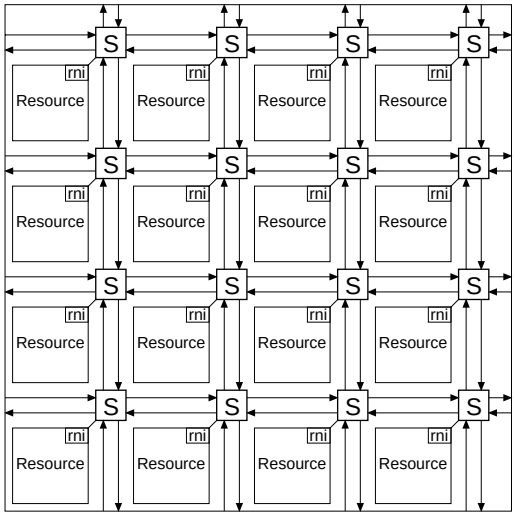

Figure 1. A NOC with 16 resources.

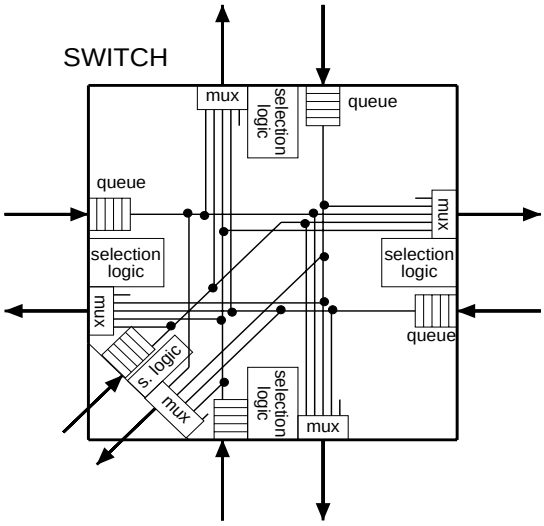

Figure 2. Block diagram of a switch.

3.2. NOC resources

The NOC would allow for arbitrary resources. Typical examples would be embedded processor and DSP cores provided with caches as well as local memories, dedicated hardware resources, and configurable hardware resources. Since the area of resource equals one synchronous clock

domain, the resource can be a combination of all previous types. The internal communication inside a resource is synchronous. In Figure 3 RNI=resource network interface, P=processor core, D=DSP core, c=cache, M=memory and re=reconfigurable block.

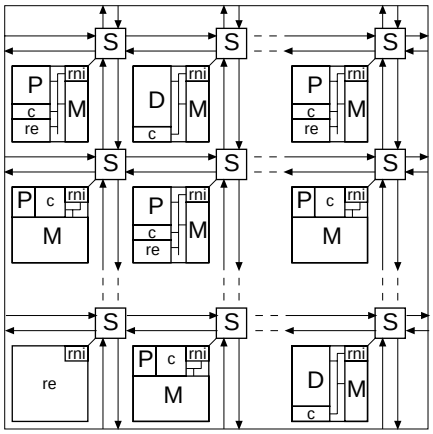

Figure 3. A typical NOC CLICHÉ featuring various types of resources.

The model of computation is a heterogeneous network of resources executing local computation. Communication between the resources is implemented by passing messages over the mesh network. Resources operate asynchronously with respect to each other. Synchronization is provided by synchronization primitives, which are implemented by passing messages around the network. Even a non-local memory is accessed through message passing.

In order to make the NOC interface with the outside world dedicated resources such as I/O elements are needed. The I/O could be of various kinds, they could glue many NOC chips together, interface with external memory or implement a TCP/IP interface. Interface modules also handle data buffering and packet reordering.

3.3. Communication

Every resource has a unique address and is connected to a network via a switch. It communicates with the switch through a RNI. Thus, any resource can be plugged into the network if its footprint fits into an available slot and if it is equipped with an RNI. The NOC defines four protocol layers:

1. The *physical layer* determines the number and length of wires connecting resources and switches.
2. The *data-link layer* defines the protocol to transmit a *cell* between a resource and a switch and between two switches. Both, the physical and the data link layer are dependent on the technology. Thus, for each new technology new technology generation these two layers are defined. Let w be the number of wires in the physical layer and c be the cell size of the data link layer. We expect that $c=n(w-w_c)$ with n=1,2,3 or 4. For n=2,3 or 4 the channel would be pipelined, accommodating n data link cells at any time instant. w_c is the number of control wires required by the physical layer, e.g. synchronization signals.
3. The *network layer* defines how a *packet* is transmitted over the network from an arbitrary sender to an arbitrary receiver directed by the receiver's network address. This layer is again technology dependent and each network layer packet, together with the destination address, is exactly 1 data link cell. Thus, taking up our previous example, we have w=300 and c may be 290. We need roughly 10 bits for the address and a few control bits (e.g. a hop count) for switching. Hence, the network packet would be 256 bit.
4. The *transport layer* is technology independent. The transport layer message size can be variable. The RNI interface has to pack transport layer messages into network layer packets.

The RNI implements all four layers towards the network. The switch-to-switch interfaces implement only the three lower protocol layers. The basic communication mechanism envisioned among computing resources is message passing. However, it is possible to add additional protocols on top of the transport layer to provide for instance a virtual shared memory abstraction, which will help the programmers in development of data and computation intensive application.

3.4. Regions and wrappers

A 2-D mesh topology provides access to all resources of the NOC, it is scalable and it has a simple structure. However, there are applications for which CLICHÉ structure is not suitable for performance reasons. Examples can found from parallel computation, digital signal processing and data flow processing areas.

A region G is an area inside the NOC, which is insulated from the network and which may have different internal topology and communication mechanisms. The concept of region allows for resources of larger size than the atomic slots in the mesh. In this way development, management, communication and instantiation concerns of various regions can be separated. Regions are connected to the NOC by special communication arrangements called wrappers W, which route packets so that regions are insulated from external traffic. Specific IO wrappers W_{io} allow communication between the region and its environment. It is also responsible for converting the messages into appropriate format. Thus, the region concept in NOC can be seen to address four aspects:

1. A region can be used to *dedicate* a set of resources and a part of the network to a specific task like processing of streaming-oriented data, processing of block-oriented data or parallel processing.
2. One can *arrange* communication inside a region differently than in the other regions. A NOC designer may e.g. want to define a region with high

communication capacity for efficient work-optimal implementation of shared memory abstraction [20].

3. A region can be used to *insulate* a set of resources from the traffic happening between the resources not belonging to the region.
4. A region can be used for *encapsulating* a specific technology into a NOC. For example an area dedicated to FPGA or embedded memory could be larger than the area of resource.

However, the shape of regions cannot be arbitrary but their boundaries must be convex. This definition of regions imply that resources requiring high-capacity intercommunication need to be placed into the same region, because wrappers between regions may cause some constraints to capacity and latency of communication. From the point of view of the network layer, regions do not form separate sub-networks, instead they can be considered as just lightweight mechanisms to organize communication in a more efficient and rational way.

4. Backbone-Platform-System Methodology

Our NOC concept is based on the idea to have a backbone based application specific platform where the final applications can be mapped as software or configurable hardware. Combination of design productivity and system quality requirements has led us to *the backbone-platform-system design methodology* (BPS). The idea with the BPS is to encapsulate the design work into reusable platforms. A NOC based system consists of a hierarchy of structural and behavioral objects, e.g. backbone, platform and system concepts. BPS has two main phases, platform development and application mapping, as depicted in Figure 4.

Even in a small 4x4 meshes of switches and resources there are 16 subsystems with a complexity of current state-of-the-art SOC design each. Management of such complexity must be based on extremely structured architecture and extensive reuse. In BPS methodology the generic, structured architecture and system development principles are described as a backbone concept. Development of several SOC complexity level subsystems, e.g. resources in CLICHÉ topology, must be based on the reuse of optimized virtual components or even computer systems. If we assume that current SOC design has a moderate complexity of 10 million gates, then even in small 4x4 mesh the hardware complexity would approach 200 million gates.

The computational capacity of NOC based system depends on the type of resources. If we assume that resources are general-purpose processor based computer systems with a capacity of 1000 MIPS each, the 4x4 mesh would have a total capacity of 16 GIPS. In real system, part of the capacity would be wasted due to communication and allocation problems, but it is obvious that reuse of applications, middleware and system architectures is required.

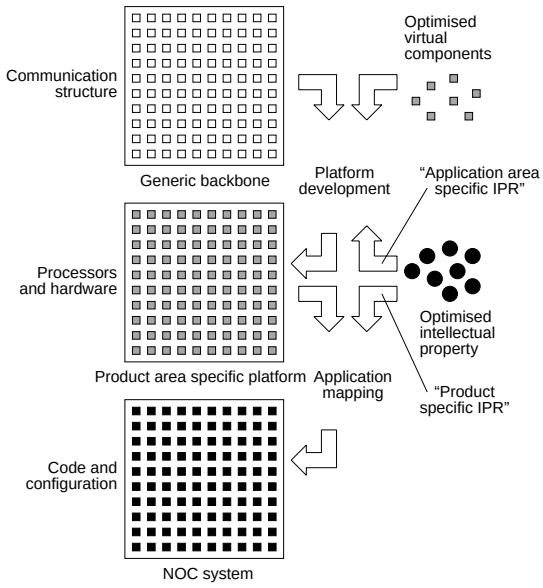

Figure 4. NOC based system design.

4.1. Backbone design

The NOC backbone encapsulates the topological and communication issues such as channels, switches, and network interfaces. The backbone is the development platform for all NOC based systems, so it is important that every system follows the basic operation principles defined in the backbone.

During the backbone design the focus is the network communication resources, e.g. switches and interfaces, and NOC system services and performance of different region topologies. From the definition of resource area follows that the connections between neighboring switches and the switch design are issues where physical design has an important role. The system-level communication challenges the technological limits. The amount of wires, wire lengths, synchronization, and buffering are all problems were physical layout and characteristics sets constraints. Customized region topology enables NOC based systems were the quality of the application mapping is optimized in the beginning. Definition of region requires that potential applications are analyzed and modeled. Mathematical and performance analyses and even performance simulations are the main tools to be used.

4.2. Platform design

The objective of platform development is to create a computation platform for an intended application area. Scaling of the network, definition or regions, design of the resource nodes, and definition of the system control are the main activities. It requires thorough understanding of the functionality of the target systems, but due to the platform

nature it is not possible to use exact applications as a starting point for architecture requirement definition. Use of optimized virtual components and knowledge of application-area requirements are essential in managing the complexity and performance requirements of the target system. During the development the characterization of application area domain and architecture and system quality estimations are essential tools. The application area specific platform encapsulates the hardware design problems and serves as a manufacturing integration platform for system developers.

For example, in 4x4 mesh CLICHÉ system, we have to define and design 16 resources, e.g. 16 communicating computer systems, if the NOC platforms would be used for the parallel implementation of heterogeneous applications. If we want to optimize the platform for some specific application area, we certainly need very efficient ways of making the right decisions and new figures of merit to describe the quality of NOC. Currently used metrics: performance, utilization, capacity must be adapted to handle temporal and spatial effects that are inevitable with target systems. For example with combined communication and computing systems, the required architectural features may vary from bit-based processing to parallel manipulation of huge data sets. The communication throughput and latency requirements are different in the same way.

4.3. System design

In the application mapping the functionality of application is mapped to the resources. The NOC concept should ultimately support both dynamic and static mapping of applications, but the main problems with both are the resource allocation, optimisation of network usage and verification of performance and correctness. Basically these issues are rather similar to what distributed and parallel system designers have to face.

The proposed NOC platform is very heterogeneous. The resources can vary from configurable hardware to multiprocessor computers of almost every type. Therefore, several modeling languages should be supported by NOC application development environment making it easy to integrate different tools into the design flow. As with platform design, the decision support and quality validation needs special attention and new approaches.

4.4. Methods and tools

Implementation of the BPS methodology or any other design flow for NOC systems will be a challenge for EDA industry. The traditional SOC, platform, and intellectual property based design flows must be extended to cover network-related issues, e.g. distribution and parallelism effects as described in Table 1.

Table 1. Design responsibilites during different phases of NOC development.

Instance	Responsibilities during design
Backbone development	Region types Communication channels and switches Network interfaces of resources Communication protocols (specification)
Platform development	Region scaling Resource design (units, interconnections) Dedicated hardware blocks System level control (implementation of communication, diagnostics, monitoring)
Application development	Resource level control (OS) Functionality of resources (SW, configurable HW) Control of the network Functionality of the network

Our NOC backbone defines the implementation of the network. The main task for designer at system level is to decide what to put into the NOC as resources, how to map functionality into those resources, and how to validate the decisions. The actual design relies on the reuse of virtual components and intellectual property, and enhanced methods and tools to support them are required. Especially at system level it is important to use abstract models and descriptions of both resources and applications. Otherwise the computational complexity of analyses, estimations and simulations will exceed the computational capacity of design tools. In traditional system design approaches the design space exploration has been done using with analytical approaches or with similar design methods and tools than the actual design. Most often, only the abstraction level of system models has been different.

In NOC design, we propose a clear distinction between decision making support, development and verification methods and tools. The *decision environment* should include methods for advanced complexity estimation, resource selection, and network analysis. Complexity estimation is needed for the scaling of NOC and for region type selection. The characteristic of computation is one issue that needs to be added to operational complexity. In the resource selection the mappability of algorithms and architectures is one alternative extension to currently used performance metrics that could provide more knowledge on the potential quality of the system. Similar analysis could be used during application mapping. Analysis of network behavior is a critical part of region definition and allocation of resources to functions. Modeling of network behavior, workload characterization and efficient simulation are the potential methods, if adapted to NOC concept. The *development and verification environments* should provide a virtual machine and development environment for software development, and tools for hardware design. Complexity is the biggest challenge in both. Abstraction, partitioning of problems and distribution of computation looks as viable alternatives.

5. Discussion

Design of a new product using NOC architecture is similar to the problem of designing a computer network with some computing and communication requirements. We have adapted ns-2 from Univ. of Berkeley at California, to study various design options in NOC architecture and their effect on performance[17].

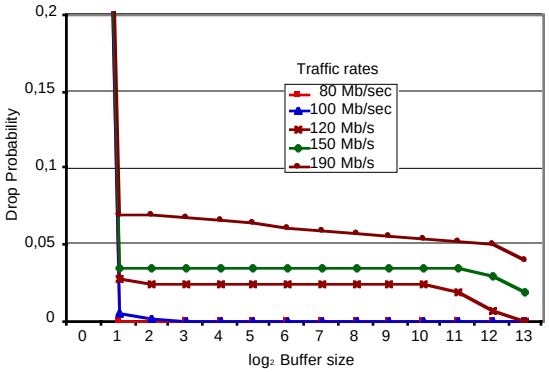

Figure 5. Drop probability vs. buffer size.

We have used a homogeneous 5 x 5 NOC architecture for our simulation experiments. In particular, we have studied the effect of buffer size in switches and network traffic (called network load) on delay and probability of message loss. These simulation experiments have been carried out using various types of network traffic cases like random traffic and local traffic and mix of these. The figure below shows relationship between the probability of a packet being dropped verses the size of buffer in the switch for each direction. We have assumed that a link between two switches supports a maximum traffic of 200Mbits/sec. Various lines in the graph show the drop probability verses buffer size for various actual traffic rates. We observe that for actual traffic of up to 100Mbits/sec, the drop probability is very close to zero if a buffer size of four packets is used. Traffic rate is controlled by controlling the rate at which a subset of resources generate packets and by controlling the destination address of the generated packets. The traffic generated for this study had a mix of local and random traffic. We have carried out many other similar experiments [17].

These simulation studies have resulted in many interesting conclusions: For moderate traffic, a buffer size of 8 messages for each direction leads to almost zero drop probability. Message delay increases with buffer size as well as network load. Message delay is more sensitive to network load than to buffer size. If the network load increases beyond 50% of network capacity, then it is impossible to avoid message drop even with large buffers.

This study helps us to decide size of buffer in switches. It also emphasizes the need for good mapping of applications to the NOC architecture so that the resulting traffic is local to a small area of the NOC. This will reduce network traffic.

5.1. Physical Aspects of NOC

We have investigated some physical issues in the design of the switches and the inter-switch connections for on-chip communication networks like NOC [18]. In particular, we have compared two distinct layouts for a switch, called “thin switch” and “square switch”. In thin switch, the switch functionality is distributed around a resource and wires are routed across the resources. A square switch is placed on the crossings in dedicated channels left between resources. The wires are routed in these channels.

We have considered wireability, delay and maximum signal bandwidth between switches, positioning of pads and positioning of repeaters in our study. The study has been conducted based on the 60nm CMOS technology expected in about 6 years. The main conclusion is that in five years 10 x 10 NOC architectures will be feasible. It will be possible to route 256 wires between a resource and a switch and between two neighboring switches in the mesh. The study also shows that the square switch option is superior with respect to performance and bandwidth while the thin switch requires relatively low area.

5.2. System development

The main objective for the NOC development environment will be to separate different concerns and activities and to shield some tools and design tasks from details in other tools and tasks.

The BPS methodology tries to benefit from reuse as much as possible and to give support for application development. The idea has also been to find an optimal balance between manufacturing and system level integration platforms. The role of the backbone is to provide a solid starting point for ASIC design with guidelines and flexibility.

The NOC system development environment will provide layered system services, which will shield an application developer from the details of the NOC lower level architecture. It will provide application level communication, synchronization, memory management, and resource management services.

Design tools, which map applications onto the NOC, must eventually implement all communications between resources by means of the three protocol layers provided by the network. This can be considered as a contract. If the applications comply with these protocols the network guarantees the communication services. Ideally we would like to extend this contract also to performance issues, for instance with a contract where applications guarantee a maximum number of messages per time unit and the network guarantees a maximum transport delay of all messages. It is part of our future work to define the conditions under which such a contract is feasible.

6. Conclusions

In this paper we have described an architectural template, called network on chip architecture, for developing large and complex systems on a single chip. The architecture supports physical level and architectural level design integration. Basic communication mechanism between resources is envisioned to be packet switched message passing through the switches. NOC architecture defines four layered inter-resource communication protocol (physical, data-link, network and transport layer), which are adapted from OSI standard. These protocols must be implemented in the resource to network interface (RNI) for every resource in NOC. We have also described a two-phase design methodology for developing systems for the proposed NOC architecture.

The NOC concept has been necessitated by three factors: First there is the increasing demand of on-chip interconnect bandwidth. The second equally crucial factor is to amortize the enormous engineering cost involved in designing such large chips over multiple applications. The third factor is demand for easy-to-use methods to exploit the parallel processing capacity provided by multiple computational resources. Programmable interconnectivity and efficient implementation of shared memory abstraction are keys to provide this generality.

Before NOC architectural template can be used to develop applications, one needs to work out the details of architecture, communication, design flow, and system services. Currently we are building many simulators for evaluating various architectural and communication options at different levels. We are also interested in analytical analysis of architectural options for NOC.

7. Acknowledgements

We gratefully acknowledge many valuable discussions we had with Dr. Li-Rong Zheng and Dinesh Pamunuwa. This work is a part of the joint Finnish-Swedish EXSITE (Explorative System Integrated Technologies) research program. This work was sponsored by TEKES (The National Technology Agency of Finland), VINNOVA (Swedish Agency for Innovation Systems), Nokia Oyj, Ericsson Radio Systems AB, and Spirea AB Kista.

References

[1] Semiconductor Industry Association, *International Technology Roadmap for Semiconductors*, World Semiconductor Council, Edition 1999, 1999.

[2] D. Sylvester and K. Keutzer, "Getting to the Bottom of Deep Submicron", *Proc. of the Int. Conference on Computer-Aided Design*, 1998, 203-211.

[3] D. Sylvester and K. Keutzer, "Getting to the Bottom of Deep Submicron II: a global wiring paradigm", *Proc. of the 1999 Int. Symp. on Physical Design*, 1999, 193-200.

[4] C. Szyperski, *Component Software: Beyond Object Oriented Software*, Reading, MA, ACM/Addison Weseley, 1998.

[5] D. Gajski, R. Dömer and J. Zhu, "IP-Centric Methodology and Design with the SpecC Language" in *System Level Design*, Edited by Ahmed A. Jerraya and Jean Mermet, Nato Science Series 357, 1999.

[6] F. Vahid and T. Givargis, "Platform Tuning for Embedded Systems Design", *IEEE Computer* 34, 3.

[7] K. Keutzer, S. Malik, A. Newton, J. Rabaey and A. Sangiovanni-Vincentelli, "System Level Design: Orthogonolization of Concerns and Platform-Based Design", *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems* 19, 12 (Dec. 2000).

[8] W. Dally and B. Towles, "Route Packets, Not Wires: On-Chip Interconnection Networks", *Proc. of the Design Automation Conference*, Jun. 2001.

[9] D. Wingard, "MicroNetwork-Based Integration of SOCs", *Proc. of the 38th Design Automation Conference*, Jun. 2001

[10] M. Sgroi et. al., "Addressing the System-on-a-Chip Interconnect Woes Through Communication-Based Design", *Proc. of the 38th Design Automation Conference*, Jun. 2001.

[11] A. Hemani, A. Jantsch, S. Kumar, A. Postula, J. Oberg, M. Millberg, and D. Lindqvist, "Network on Chip: An architecture for billion transistor era", *Proc. of the IEEE NorChip Conference*, Nov. 2000.

[12] A. Jantsch, J. Soininen, M. Forsell, L. Zheng, S. Kumar, M. Millberg, and J. Öberg, "Networks on Chip", *Workshop at the European Solid State Circuits Conference*, Sep. 2001.

[13] L. Benini and G. DeMicheli, "Powering Networks on Chip", *Proc. of the 14th Int. Symp. on System Synthesis*, 33-38, Oct. 2001.

[14] F. Catthoor, D. Verkest, and E. Brockmeyer, "Proposal for unified system design meta flow in task-level and instruction –level design technology research for multi-media applications", *Proc. 11th Int. Symp. on System Synthesis*, 1998, 89-95.

[15] P. Panda, N. D. Dutt, and A. Nicolau,, "Local Memory Exploration and Optimization in Embedded Systems", *IEEE Trans. on Computer Aided Design of Integrated Circuits and Systems* 18, 1 (1999), 3-13.

[16] A. Hemani et. al. , "Lowering power consumption in clock by using Globally Asynchronous Locally Synchronous Design style", *Proc. of Design Automation Conference*, 1999, USA.

[17] Yi-Ran Sun, "Simulation and Performance Evaluation for Network on Chip", *MSc thesis*, Dept. of Microelectronics and Information Technology, Royal Institute of Technology, Stockholm.

[18] Dinesh Pamunuwa et. al., " A study of Physical Issues in the design of an on-chip regular communication network", *submitted to DAC 2002*.

[19] V. Leppänen, Studies on the realization of PRAM, *Dissertation 3*, TUCS, University of Turku, 1996.

[20] M. Forsell and S. Kumar, Virtual Distributed Shared Memory for Network on Chip, *Proc. of the 19th IEEE NORCHIP Conference*, Nov. 12-13, 2001, Kista.