Extending Platform-Based Design to Network on Chip Systems

Juha-Pekka Soininen[1], Axel Jantsch[2], Martti Forsell[1],
Antti Pelkonen[1], Jari Kreku[1], and Shashi Kumar[2]
[1] *VTT Electronics (Technical Research Center of Finland), P.O. Box 1100, 90571, Oulu, FINLAND*
[2] *Laboratory of Electronics and Computer Systems, Department of Microelectronics and Information Technology, Royal Institute of Technology, 164 40, Kista, Stockholm, SWEDEN*
E-mail: Juha-Pekka.Soininen@vtt.fi

Abstract

Exploitation of silicon capacity will require improvements in design productivity and more scalable system paradigms. Asynchronous message passing networks on chip (NOC) have been proposed as backbones for billion-transistor ASICs. We present a novel layered backbone-platform-system (BPS) design methodology for development of network-on-chip based products. It combines and extends the distributed, parallel, embedded and platform-based design concepts in order to manage the diversity and complexity of NOC-based systems. The reuse of communication principles in various platforms, the reuse of platforms in product differentiation, and system-level decision-support methods are the cornerstones of our methodology. The presented mappability estimation and workload simulations demonstrate the feasibility of such methods.

1. Introduction

Rapid technology development is expected to continue even without any revolutionary inventions [1]. It will be possible to put about one hundred embedded computer systems on a single chip within ten years. Those embedded systems may have multiple processor cores and coprocessors totaling to almost one thousand computing architectures. The computational capacity of such a system could reach 1000 GOPS even with moderate clock frequencies and reasonable power consumption. Applications would consist of tens of thousands of functions that would further need billions of operations when executed.

Platform based design is a proposed solution for system on chip (SOC) complexity problems. The existing design or model is used as a platform for the development of next abstraction level. Different types of platforms ranging from tool environments to implemented product platforms have been proposed [2]. Recently, the focus has been on providing a solid basis for system configuration and software development [3, 4]. In the platform based design approaches, the target has been a complex embedded system with one or few processor cores. However, instead of a single system, an integrated network of more or less autonomous systems will be needed [5, 6, 7, 8]. Physical limitations, asynchronous solutions and message passing based on-chip communication will be in important role.

Two principal approaches for using available silicon capacity are following.

1. Parallel, general-purpose computer has a simple programming model. It provides superior performance in case the application characteristics vary a lot or if there is lot of communication between different system functions.
2. Application specific computation engine results to benefits in energy optimization and subsystem performance in case the application can be distributed to more optimal subsystems. This is a valid assumption in telecommunication and multimedia applications.

We present a network on chip (NOC) design methodology that extends the platform based design approaches to the systems with on-chip networks and heterogeneous computing resources. We claim that the future system development methodology has to integrate existing design methods and to provide means for dealing with different types of models and subsystems. The novelty in our approach are in emphasizing the role of decision-making support especially on early design phases and in tight integration with physical level and architecture level issues. The methodology partitions the development work into abstraction layers, application domains, and design activities. The partitioning is made possible by the structural approach to the NOC architecture. The main challenge is the complexity that requires using more abstract models in decision-making.

The performance analysis and evaluation is the essential part of computing system design [9]. Techniques that are faster than traditional processor simulations are needed for exploring the feasibility of system components [10, 11]. Two examples of methods are presented in this paper in order to demonstrate the type of methods needed in our approach. A novel mappability estimation method introduces a new measure of goodness for the evaluation of the quality of processor core and algorithm pair. The method can be used in the very initial phases of system development because it does not require executable specification or simulation model of the processor. The performance validation approach for multiprocessor system presented in the paper uses workload models and abstract architecture models resembling the ones used in processor performance simulations.

Structure of the paper is following. Chapter 2 introduces the NOC concept, and its quality criteria. Chapter 3 introduces our design methodology for NOC platforms and systems. Decision support methods for NOC are described in more detail in Chapter 4. Chapter 5 gives our conclusions.

2. Network on chip

Network on chip (NOC) is a new paradigm for large SOCs trying to solve the design productivity, usability and architectural problems related to future SOCs. A uniform communication network connecting on-chip resources and providing much better bandwidth scalability than buses differentiates NOCs from SOCs. The main problems of NOC are area and performance overheads in respect to optimized dedicated hardware solutions.

2.1. Quality characteristics of NOC based systems

As any other electronic system, NOC based systems have to meet a variety of requirements, including correct functionality, sufficient throughput, latency and memory, high energy efficiency, maintainability and fault tolerance. In the sequel we sometimes denote all these various characteristics with the term "performance". However, in a NOC based approach we distinguish between responsibilities between the NOC platform and the application designer to meet given requirements. A major advantage of any platform-based approach is the provision of guaranteed performance characteristics formulated at an abstract level, which can be conveniently used by application designers. For instance, a traditional CPU would guarantee that any sequence of instructions could be executed with a particular frequency. This simple and general assurance can be easily used by application programmers and compilers without worrying about wire delays, noise margins and specific corner cases. Correspondingly, a NOC based platform should guarantee performance figures for communication, reliability levels of the communication, and power efficiency levels of the communication.

Thus, we have to distinguish between the quality characteristics provided by the NOC platform and the quality characteristics achievable by an application designer when utilizing the platform. In defining the NOC platform quality characteristics we have to consider three main directions:

Communication infrastructure: It has to be provided together with performance guarantees for delay, bandwidth, power consumption, and communication reliability. The basic communication functionality has to be accompanied by performance figures to allow application engineers to build reliable applications with predictable performance. The performance figures may sometimes be expressed in stochastic terms and may vary between application and traffic types, but altogether they must be sufficiently transparent and simple to be usable at higher levels.

Flexibility: Since the main point of a platform is to support a variety of concrete products and applications, flexibility is a major concern. Hence, the platform has to support different traffic types with widely varying requirements. Safety critical systems will put higher demands on predictability than multi-media applications. Power management and efficiency is of highest concern in hand-held devices. The matter is complicated further by the prospect, that various application types will coexist in future NOC based products. Thus, a NOC platform has to be careful at offering a sufficient range of choices to application engineers.

System integration: A methodology has to provide sophisticated means to for system analysis. Since the final product will contain a large number of different resources it will be a formidable task to integrate the performance assessments of the resources with the performance guarantees provided by the network into a reliable system performance assessment. Most of this paper focuses on this particular issue.

2.2. NOCARC approach

Our approach for NOC is a scalable application area specific computer network providing packet switched platform for future SOC [12]. The architecture is an m x n mesh of switches and resources are placed on the slots formed by the switches. The diameter of the slot is limited by the clock cycle. So, each resource forms a synchronous clock domain, but the communication between resources is implemented with asynchronous messages. 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 network interface. The 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 multiprocessors.

Many architectural solutions in our NOC are results from physical-level issues. These physical-level and architecture issues affect also to our design methodology. On the other hand, the design productivity requirements and product functionality requirements that have been set to design methodology affect to architectural solutions. Our NOC can be seen as an integrated distributed embedded system. It is integrated because it is implemented on a single chip. Its application functionality is distributed to resources and each resource is an embedded system type of SOC. Software development resembles embedded system software development and the resource design is a SOC design type of a problem.

Our approach provides improvements to simple on-chip computer network via the region concept, the layered communication and the possibility to dedicate resources (such as embedded FPGA or memory or reconfigurable fabric of processing elements). The main benefits are power saving possibilities, locally optimal execution of functions, relatively simple programming model and reuse of existing IP blocks.

3. NOC Platform Methodology

The success of a NOC depends on how effectively it can satisfy the often-contradictory requirements of applicability and performance. Applicability requires flexibility and generality, simple programming models, and simple development of final functionality. Performance requires efficiency, power awareness, dedicated structures, and optimal combinations of resources and functions.

It is obvious that a NOC design methodology must be based on heavy reuse of existing designs, reuse of software systems and reuse of platforms. The whole NOC concept is based on this idea of encapsulating embedded systems into a network, computers, reconfigurable fabrics, or embedded memories into resources, and intellectual property blocks, virtual components and software into embedded computer systems. The hierarchical encapsulation introduces well-defined interfaces between different layers of NOC system and allows constructing the complete system from black-box type of units.

Design methodology means partitioning of the problem to manageable tasks and definition of tools and design practices for those tasks. We have partitioned our design methodology in three different ways. Firstly, we have divided the NOC system development in three main abstraction layers that are backbone, platform and system development. Secondly, we have divided the methodology into application domains according to what kind of a subsystem we are developing. Thirdly, we have divided each design activity into phases according to the purposes of those phases. Every design activity should compose of analysis, estimation, decision and validation activities.

3.1. Layers in NOC development

The motivation for layered NOC development is to separate technology specific e.g. backbone issues, application area specific e.g. product platform issues and product instance specific e.g. system issues from each other. The main layers and how they relate to NOC system development are shown in Figure 1.

NOC backbone layer: The NOC backbone consist of physical components required for communication such as channels, switches, and network interfaces, and basic services required by the communication such as physical and data link layer protocols. The focus is in the network communication resources, e.g. switches and interfaces, and NOC system services and performance of different region topologies. The physical design issues have an important role in the design of the communication channels and switches because the system-level communication challenges the technological limits. The wires, wire lengths, synchronization, and buffering are constrained with physical layout and implementation.

NOC platform layer: The platform is a computation platform for target application area. Platform design requires the understanding of how the target systems operate and what kind of computation they have. Scaling of the network, definition or regions, design of the resource nodes, and definition of the system control are the activities that must be based on abstract models of applications. The platform encapsulates the hardware design problems and serves as a manufacturing integration platform for system developers.

NOC system layer: The NOC system is programmed chip in the final product. The resource allocation, optimisation of network usage and verification of performance and correctness are the main problems that are basically similar to what distributed and parallel system designers have to face.

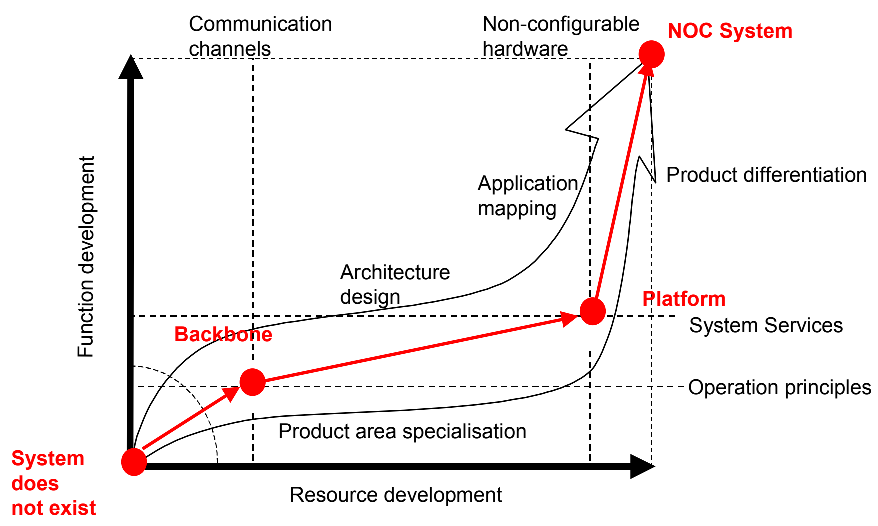

Figure 1. NOC design flow with backbone, platform and system phases. Backbone encapsulates communication resources and operation principles. Platform contains the non-configurable hardware resources and system services.

3.2. Design methods for NOC resources

The proposed integrated distributed embedded system concept says very little about the contents of embedded resources. There are some obvious constraints. They must be implementable with backbone technology. They must implement the system services required by the platform such as power management, diagnostic and testability functions. Otherwise, the platform designer can use all his creativity to improve the applicability of the platform in the application area domain.

Freedom of choices is a challenge for design methodology. Consistent methodology must provide tools and techniques that allow designing and modifying everything that are not fixed. The resources in our approach are almost autonomous subsystems and there exists lot of methodologies that can be used during their design. Therefore, we have organized our methodology as a hierarchy of subsystem specific methodologies. The lower level methodologies can be used in implementing the subsystems of upper levels, as long as upper level constraints are fulfilled. So, we can reuse existing and widely used methods and tools. The methods at each level are following:

1. System-level design and networked system's design methods. System-level design deals with application modeling and analysis, while network design methods are applied for mapping and load balancing.
2. Parallel computer system design methods, distributed computer system methods, embedded system design methods, etc. The selection of used method depends on the type of region that we are developing.
3. Computer system design, software system design, codesign, or configuration design depending on the resource, are examples.
4. Basic software, logic and synthesis methods and design flows.

The feasibility of such a hierarchical methodology depends on how easy it is to model and present the constraints for lower levels and how much these constraints actually effect on the final system. Platform based design, the reuse of intellectual property, standardized interfaces such as application program interfaces and virtual component interfaces are essential elements of successful implementation of the complete methodology. Integration of design information has become easier because of the achievements in SOC research community and in co-operation of main companies and EDA vendors. Virtual Socket Interface Alliance, Virtual Component Exchange and Open SystemC Initiative are good examples of efforts that increase the design productivity.

3.3. Design phases in NOC

Our methodology divides all design activities into four phases that are analysis, estimation, decision, and validation.

Analysis phase is needed for understanding what we have as input for our design decision. For example, when software system is designed the first task is requirement analysis, where the objective is to find out what are the user needs that should be implemented but implementation considerations are strictly forbidden.

Estimation is part of design space exploration where to objective is to forecast the outcome of possible decision without putting too much effort into the modeling of the decision.

Decision is the phase when selections are implemented as a new system model. Decision includes the creation of a new model, which includes the refinements and transformations.

In the validation phase the effects of decisions are measured using the new system model as input. The simulation or formal verification is part of the problem, but the validation should consider all the quality characteristics of the system

The actual content of each phase naturally depends on the NOC method and layer. Table 1 lists the most important issues that must be considered in main layers and phases in NOC system design.

Table 1. Important issues at different NOC design phases and layers.

	Backbone layer	Platform layer	System layer
Analysis phase	Silicon characteristics	Application workload	System modeling
Estimation phase	Yield Power Area	Capacity Efficiency Performance	Feasibility Utilization Performance
Decision phase	Channels Switches Protocols	Resources Services	Mapping Design
Validation phase	Performance analysis	Performance simulation	System Simulation

The partitioning in our methodology gives plenty of possibilities for NOC-based system designers. Physical partitioning to regions and resources allows reusing even complete computer systems as long as they follow the interfacing rules. It is also possible to allocate some resource slots to highly optimized architectures or subsystems, and the domains allow to design them using most appropriate methods. Optimized subsystem architectures result to power saving and performance benefits, and the network structure even allows dynamic controlling of the activities of resources. The partitioning of design phases enables separation of concerns and makes problems manageable.

4. Decision support methods for NOC

Advanced decision support methods and tools are necessity for NOC development. Both platform and application development have large number of alternative solutions, which can not be studied in detail. We have to limit the design space very rapidly without excluding potentially good alternatives, which means that we have to use coarse-grained evaluations and estimated figures of merits (FOMs). Attractive choices for FOMs are the quality characteristics presented earlier, but the number of possibilities is large, and it is not feasible to spend too much time and resources on estimations either. The most essential estimations from the NOC perspective are related to mappings between physical objects and functionality, and to the performance of the system.

4.1. Mappability estimation

The mapping of functionality into resources has huge impact on performance, power consumption, functionality, cost and variability of the system. In NOC systems the mappings are done in several abstraction layers.

First the functionality is mapped into a NOC region. It may be necessary to identify sets of applications that are effectively implemented with a specific type of architecture such as parallel computer or dedicated memory organization. The evaluation of such mappings is closely related to complexity estimations of different application characteristics, but the estimations must take into account the characteristics of region also.

Secondly, the functionality must be mapped to resources e.g. SOCs in the NOC. The problem is the same as with distributed systems, with the exception that distribution is not caused by physical constraints, but because of efficiency, power consumption, performance, etc. reasons.

Thirdly, the smaller parts of functionality must be mapped to the functional units in SOC e.g. processors and coprocessors. SW/HW or function/architecture codesign deals with this problem and there are some solutions especially if functionality and target architecture are known. In the NOC platform development the applications are not known in detail and the target is not to find optimal combination but to specify suitable resources for variety of applications. During the NOC system design the objective is to find most optimal processor from fixed set of alternatives and designer's freedom is limited to a modification of algorithm.

The evaluation of the quality of processor core and algorithm pair supports rapid exploration of design space and it can serve as a basis estimations at upper layers of NOC too. We have developed a mappability estimation method, in which we create a set of estimates on both algorithm and core characteristics and compare them in order to see how well the correlate [13]. An abstract control-data flow graph CFDG with branch probabilities and data value bounds is generated from C-model using SUIF2 compiler and profiling tools. The architecture model consists of basic architecture parameters such as superpipelining degree, number of parallel execution paths, number of registers and buses, performance parameters such as branch prediction efficiency and data bypassing efficiency, and an instruction set model.

The correlation is divided into six parts that consider separate aspects of program execution. Instruction set correlation examines how effectively and extensively the core instructions are used. In data flow continuity correlation we study how much data dependencies cause data hazards and degrade the pipeline efficiency. Control flow continuity correlation depends on the number of branch instruction and superpipelining degree. Taken branches decrease the overlapping possibilities provided by architecture. The execution unit availability correlation compares operation level parallelism to the number of parallel execution units. The parallelism of algorithm is constrained by the data dependencies and it can be studied from the number of operations in each step in the unconstrained ASAP schedule. Data availability correlation expresses how effectively registers and buses can be used. The number data dependencies inside basic blocks should correlate with the number of available registers and the number of data dependencies across basic boundaries should correlate to bus capacity.

We have done experiments with processor core selection for Hiperlan/2 modem. The problem is similar to what we face during NOC platform development. We wrote C models for 13 baseband functions of HiperLan/2 modem. Viterbi encoder and some trivial functions were excluded from the study because the hardware implementation was obvious. C models were converted to CFDG presentations and mappability estimates were calculated. The database of possible processor core architectures had 10800 different architectures that were generated by varying the architecture parameters. The instruction sets were not considered in the evaluation. 140400 mappability estimates were analyzed in order to find the best architectures for individual functions and for feasible software implementation. As a results we had mappings that are relatively close to best possible mappings with three core architectures. The architecture for parallel functions had longer superpipelining degree and more registers. Two architectures for more control-dominated functions were simpler. Their main difference was in bus capacity. [13]

Our mappability estimation makes simplifications that may effect on absolute values of estimates. It also ignores some of the issues that are very important at the SOC level, such as cache memories and block level parallelism. But it is very fast and powerful. Its mappability metric combines performance, resource utilization, execution efficiency, and power consumption. It also considers the algorithm and architecture as a combination as they are in the final product. When the models were ready, the estimation of all 140400 mappings took only few minutes in a single PC-workstation. This kind of estimation performance is necessity with systems comprised of hundreds of processors and it cannot be achieved with simulation-based approaches.

4.2. Performance validation

The rising of NRE and design costs force us to design more versatile systems, and the time-to-market pressures demand us to design systems implementing communication and multimedia standards which are not yet finalized. Current implementation of basic GSM phone for example is very different from the first version, and in GPRS and EDGE evolution versions the complexity increases further. To validate design decisions, methods for analyzing and simulating system performance and behavior at various levels of abstraction are needed. Three different simulation levels are identified for NOC system as illustrated in Figure 2.

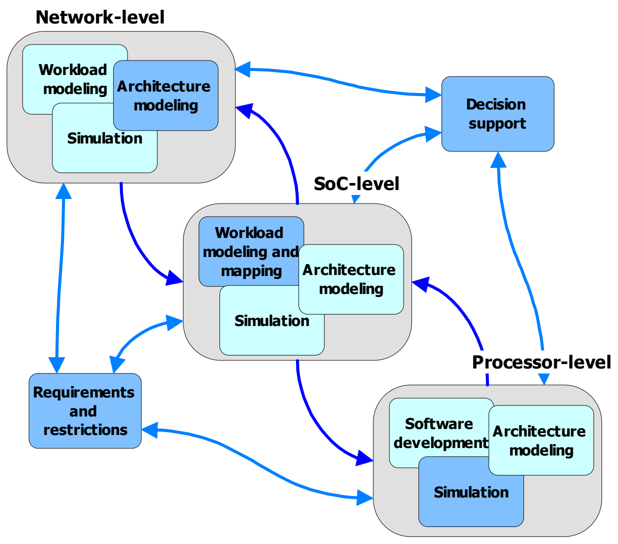

Figure 2. High abstraction level quality validation flow

The performance of the network topology and communication must be validated at *network level* with network simulation. The main parts in network simulator for NOC are resource model, switch model and model of the network structure. Our region concept can be implemented as a separate communicating network simulation, or as a resource model, if we can assume that the messaging behavior is known.

The resource model is an abstraction of the SOC. Its functions are to emulate SOC behavior, generate messages into network and to collect statistics of the messages. In order to support all NOC layers, it must be possible to link different types of functional models of SOCs as workload generators. For example, at backbone layer very simple message generator for signals is enough, at platform layer statistical model or very abstract functional model for message generation is applicable, and at system layer it may be necessary to execute even the final code.

The switch model implements the buffering and routing functions. At the design phase exploration phase it is important to be able to rapidly modify the messaging protocols, physical dimensions such as number of wires and queue sizes, and routing algorithms.

Commercial and public domain network simulators can be used for evaluating also NOC systems. We have used Matlab and NS-2 for buffer sizing, for example [12]. The tools are mostly targeted for the simulation of Internet or other standardized protocol based communication. The network simulation needed for NOC development needs more versatility and support for NOC specific monitors and analyzes that are not only network dependent. Therefore we have started to develop our own network simulator.

At resource or *SOC level* the computation capacities of processors, communication capacities of buses and storage capacities of memories must be checked. Workload modeling and transaction level simulation at SOC level is most attractive solution. Transaction level modeling means that communication of architectural units happen in transactions such as read, write or burst-read, instead of pin-level events. Simulation speed is significantly faster than with traditional coverification tools and it is also easier to integrate software models and monitors into the simulation.

We have used SystemC 2.0 that provides constructs for transaction-level modeling in form of interfaces and channels. The actual processing elements are implemented as communication shells and the functionality of applications as workload models. Initially, the purpose of the simulations is to validate the architecture but in further iterations, it can be used for balancing the utilization of architectural units.

We have experimented the SOC level simulations with a multiprocessor SOC consisting of 4 DSP processor cores, a RISC core and FFT and Viterbi-decoding co-processors. The SOC architecture was modeled and simulated with workload models of Hiperlan/2 modem transceiver [14]. The functional complexity was analyzed and resulting workload models were mapped into platform resources. Utilization of processing elements and the utilization and simultaneous memory references of shared memory bus were measured using SystemC monitors. The utilization of main elements of SOC with slowest bit-rate of Hiperlan/2 modem are presented in Table 2. As it can be seen the target architecture is not optimal for the application. For example, the memory buses are oversized and the Viterbi coprocessor is under used, since it can handle the full-bit-rate for Hiperlan/2. These simulations demonstrate the power of transaction level modeling that is a necessity in NOC system design. The simulation of 100 000 clock cycles with SystemC took less than two minutes with SunBlade 1000 workstation with two 900 MHz Ultrasparc III processors. This is significantly less than with instruction set simulation or RTL-VHDL simulation. The reliability of results depends mostly on the accuracy of workload models and it has little effect to simulation time. So, it is possible to make fast evaluations of alternatives. The SystemC model may also be re-used as test-bench when advancing to actual RTL HW design or SW design. The benefit of this approach is to have an intermediate level of abstraction between formal models and register transfer level. The software and communication design can also be started early with workload models.

Table 2. Utilization of processing elements and buses in 6Mbit/s Hiperlan/2 WLAN application

	Busy	Wait	Idle
RISC core	48%	10%	42%
DSP1 core	38%	57%	6%
DSP2 core	83%	17%	0%
DSP3 core	50%	29%	21%
DSP4 core	63%	33%	4%
FFT coprocessor	50%	0%	50%
Viterbi coprocessor	3%	0%	97%
X1_Bus	23%	4%	73%
X2_Bus	17%	1%	82%
X3_Bus	12%	0%	88%
X4_Bus	30%	1%	69%
Y1_Bus	15%	1%	85%
Y2_Bus	0%	0%	100%
Y3_Bus	3%	0%	97%
Y4_Bus	23%	2%	75%

Transaction level simulations do not validate instruction, signal, and clock level details that are needed at *processor level*. Instead of workload models, the processing element shells will be used to connect instruction set simulators (ISS) or similar to the model for being able to run actual target software in simulation model. The SOC-level SystemC model may also be used as test-bench when refining the transaction level models in to RTL hardware, since the same language can be used in simulation of RTL and higher abstraction level models. There is already limited commercial tool support for automating the building of co-simulation environment.

The described method of quality validation approaches the problem of architecture design with defining the used abstraction levels and set of tasks that must be executed when moving from one abstraction level to another. Since the design flow is iterative, first results may be generated very quickly and architecture and workload models may be improved when needed. The gap of architecture design and RTL design is narrowed, as is the gap between software and hardware design and debug environment with this approach.

5. Conclusions

When we enter the billion-transistor area, the basic structures in ASICs have to be reconsidered. The systems have to be built on the top of network concepts and extensive reuse must be exploited. Application area specific NoC is a viable alternative, because it provides higher performance with lower power consumption and cost for targeted applications.

We have presented a new design methodology for NOC system development. The methodology reuses the established design practices and extends them with network specific methods and tools. We have presented an approach that takes into account the optimization of network behavior in all design stages and abstraction layers. Our approach also exploits the best parts of platform based design e.g. the reuse of tested and validated designs and standardized interfaces to application development.

We believe that the cornerstones of the development of extremely complex integrated systems are the efficient estimation and validation methods that aim at decision support. We have done work with the estimation of mapping quality at processor core-algorithm level. The results so far are promising and we are convinced that same principles can be applied to more complex objects as well. The combination of network simulation, transaction-level modeling and coverification seems as a feasible approach for quality validation of networked systems. Our experiences are that valid results can be achieved also with abstract models and fast simulations as long as the principles and objectives of modeling are correct.

6. Acknowledgements

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.

7. References

[1] Allan, A. et al, “2001 Technology Roadmap for Semiconductors”, *Computer*, Vol. 35., No. 1, 2002, pp. 42-53

[2] Chang, H., et al, *Survining the SOC Revolution – A Guide to Platform-Based Design*, Kluwer Academic Publishers, 1999, 235 pp.

[3] Keutzer, K. et al, “System Level Design: Orthogonalization of Concerns and Platform-Based Design”, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Volume 19, Issue 12, December 2000. pages 1523 –1543

[4] Sangiovanni-Vincentelli, A. & Martin, G., “Platform-Based Design and Software Design Methodology for Embedded Systems”, *IEEE Design and Test of Computers*, November-December, 2001, pp. 23-33

[5] Guerrier, P. & Greiner, A., “A Generic Architecture for On-Chip Packet-Switched Interconnections”, *Proc. of Design, Automation and Test in Europe*, 2000, pp. 250-256

[6] Dally, W. J. & Towles, B., “Route Packets, Not Wires: On-Chip Interconnection Networks”, *Proc. of Design Automation Conference*, 2001, pp. 684-689

[7] Benini, L. & De Micheli, G., “Networks on Chips: A New SoC Paradigm”, *IEEE Computer*, Vol. 35, No.1, 2002, pp. 70-78

[8] Mai, K. et al, “Smart Memories: a modular reconfigurable architecture”, *Proc. of Int. Symposium on Computer Architecture*, 2000, pp. 161-171

[9] Bose, P. and Conte, T. .M., “Performance Analysis and Its Impact on Design”, IEEE Computer, Vol. 31, Issue 5, 1998, pp. 41-49

[10] Krishnaswamy, U. and Scherson, I.D, “ A Framework for Computer Performance Evaluation Using Benchmark Sets” IEEE Transaction on Computers, Vol. 49, No. 12, 2000, pp. 1325-1338

[11] Gupta, T.V.K, et al., “Processor Evaluation in an Embedded Systems Design Environment”, *Proc. VLSI Design, 2000*, pp. 98-103

[12] Kumar, S., et al, “A Network on Chip Architecture and Design Methodology”, *Proc. of 2002 IEEE Computer Society Annual Symposium on VLSI*, 2002, pp. 117-124

[13] Soininen, J-P., et al, “Fast Processor Core Selection for WLAN Modem using Mappability Estimation”, *Proc. of the 10th International Symposium on Hardware/Software Codesign*, 2002, pp. 61-66

[14] Soininen, J-P., et al, “Configurable Memory Organisation for Communication Systems”, *Proc. of Euromicro Symposium on Digital System Design*, 2002, pp. 86-93