Joint Validation of Application Models and Multi-Abstraction Network-on-Chip Platforms

Sanna Määttä, Tampere University of Technology, Finland

Leandro Möller, Technische Universität Darmstadt, Germany

Leandro Soares Indrusiak, University of York, UK

Luciano Ost, Catholic University of Rio Grande do Sul, Brazil

Manfred Glesner, Technische Universität Darmstadt, Germany

Jari Nurmi, Tampere University of Technology, Finland

Fernando Moraes, Catholic University of Rio Grande do Sul, Brazil

ABSTRACT

Application models are often disregarded during the design of multiprocessor Systems-on-Chip (MPSoC). This is due to the difficulties of capturing the application constraints and applying them to the design space exploration of the platform. In this article we propose an application modelling formalism that supports joint validation of application and platform models. To support designers on the trade-off analysis between accuracy, observability, and validation speed, we show that this approach can handle the successive refinement of platform models at multiple abstraction levels. A case study of the joint validation of a single application successively mapped onto three different platform models demonstrates the applicability of the presented approach.

Keywords: *Actor Orientation, Application Modelling, Multiprocessor Systems, Network-on-Chip, Platform Modelling, UML Sequence Diagram*

INTRODUCTION

The design space of multiprocessor systems based on Network-on-Chip (NoC) interconnects is very large. Considering the interconnect structure alone, designers must choose or parameterise a large number of components according to application-specific constraints, such as buffering and flow control mechanism, network topology, word width, packet structure, or routing algorithm.

In this article, we propose a model-based methodology to evaluate the performance of a

DOI: 10.4018/jertcs.2010103005

particular setup of a NoC interconnect under the constraints of a particular application. Our goal is to support the creation of a unified application model, which we can jointly validate with different platform models, each representing an alternative configuration of the NoC interconnect.

The validation of application models and platform models for NoC-based systems can be performed in several ways, ranging from formal and semi-formal approaches based on graph representations of both application and platform (Hu & Marculescu, 2005) to ad-hoc approaches that actually emulate the execution of the application on a prototype of the NoC using multiple FPGA boards (Saint-Jean et al., 2005). The next section presents further details of such approaches. Our approach combines semi-formal techniques with executable and simulatable models for joint validation of the application and platform. Both models are based on the principles of actor-orientation (Eker et al., 2003), which the Application Modelling Section covers. Actor-oriented models include the explicit description of the concurrent behaviour of the model's components. To improve the expressiveness of actor-oriented models, we extend them by using Unified Modelling Language (UML) (Object Management Group, 2005) sequence diagrams for explicit ordering of inter-component communication. Applying such technique to application modelling is also addressed in the Application Modelling Section, whose particular emphasis is on the employment of hierarchical composition of systems with heterogeneous models of time and concurrency (one of the major benefits of actor-orientation). The Platform Modelling Section covers the modelling of hardware platforms based on NoCs. Within the scope of this article, a hardware platform includes the interconnect structures, an abstract representation of processing and storage elements, and a number of observability features (referred as scopes). We present a few strategies for the modelling of such platforms, each of them having different characteristics regarding simulation speed, accuracy, observability, and modelling effort. The Application Mapping onto Successively Refined Platform Models Section presents a methodology for the exploration of those different platforms in order to meet the application requirements. Furthermore, we present a case study of the validation of the proposed methodology. Then we analyse qualitatively each modelling strategy and comparatively the performance results obtained from different configurations of the NoC interconnect. Finally, we conclude the article and present some future work.

RELATED WORK

This article is based on the premise that it is necessary to consider the impact of the application on its underlying platform early at the design process in order to meet all performance, area, power consumption, and time-to-market constraints. Many research initiatives are also built on that premise, especially in multiprocessor System-on-Chip (MPSoC) design. Some of them are detailed below.

Kempf et al. (2006) present a framework targeted to MPSoC software development, verification, and evaluation. Their framework does not require the platform model to be complete before the software development can start. Software can be developed in four different levels of abstraction that vary in accuracy and simulation speed. The framework uses SystemC for simulation and XML to describe task mappings and timings. Furthermore, the framework provides an efficient design space exploration environment for instance by providing designers with various communication architectures. Ristau et al. (2008) discuss design space exploration early at the design process as well as the exploration of different mapping strategies. However, their application and platform models are simplified, disregarding the inter-process communication costs. Lei and Kumar (2003) describe the application as parameterisable task graphs, which are mapped onto NoC architecture. Their work aims at supporting mapping based on genetic algorithms

and minimising overall execution time of a task graph. Furthermore, they have also implemented a tool that uses a two-step genetic algorithm, achieving optimised solutions for regular NoC topologies in affordable time. In contrast to the work mentioned above, Saint-Jean et al. (2007) have an ad-hoc approach that emulates the execution of the application on a prototype of the NoC using multiple FPGA boards.

UML supports object-oriented system design, which, among other things, makes it popular among software designers. In fact, its popularity is increasing also among hardware and embedded system communities (Martin, 2002). Riccobene et al. (2005) present a System-on-Chip (SoC) design methodology using code generation from UML diagrams to an executable SystemC model. Furthermore, Arpinen et al. (2006) use UML state chart diagrams to support automatic code generation in order to map UML applications onto the platform. In this article, we rely on a similar UML-based modelling approach, but we do not require code generation in order to validate the application. Instead, we provide the model itself with execution semantics so that we can validate the model directly.

APPLICATION MODELLING

Due to the complexity and heterogeneity of state-of-the art embedded systems, we need to use various levels of abstractions and multiple Models of Computation (MoC) (Eker et al., 2003; Martin, 2002). Heterogeneous modelling using SystemC (Grötker et al., 2002) would require extensions to the standard SystemC simulation kernel, so that models containing Synchronous Data Flow (SDF), Communicating Sequential Processes (CSP), or Finite State Machine (FSM) MoCs can be simulated with the alternative kernel and Discrete Event (DE) models with the standard kernel (Patel & Shukla, 2004). In order to avoid ad-hoc solutions for truly heterogeneous system modelling and simulation, we can find a suitable alternative on the Ptolemy II framework (Brooks et al., 2007), which natively supports the composition of multiple MoCs on a single system model.

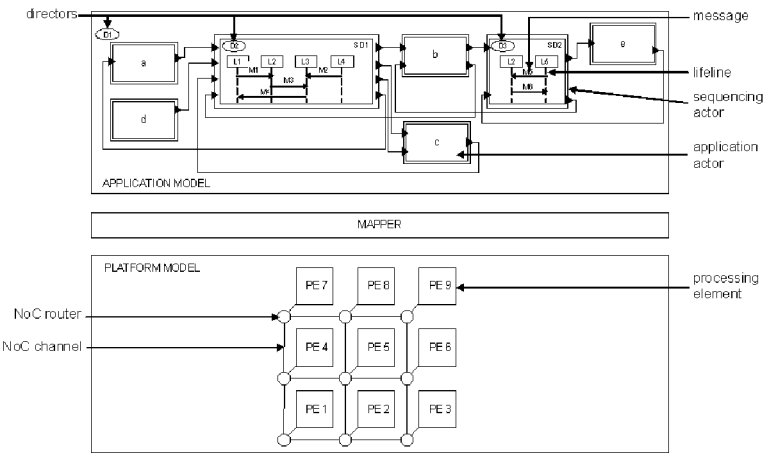
Actor-orientation, which is the theoretical foundation behind the Ptolemy II framework, was initially proposed as a mathematical model of concurrent computation (Hewitt et al., 1973) and later as a formal model of concurrency (Agha, 1986). The main components of actor-orientation are actors, which are concurrent elements and use channels and message passing for communication. The Ptolemy II framework offers visual modelling of actor-oriented designs and a simulation kernel, which supports the simulation of hierarchical designs using different MoCs for different subsystems. Furthermore, Ptolemy II framework allows the implementation of new MoCs. Using the ideas of actor-orientation and the flexibility of the Ptolemy II framework, we can create an executable application model using UML sequence diagrams for describing the communication patterns of the application and encapsulating the diagrams inside an actor (we call it a sequencing actor) (Indrusiak et al., 2007; Indrusiak & Glesner, 2007). The upper part of Figure 1 shows an application model, particularly illustrating the encapsulation of UML sequence diagrams inside sequencing actors.

Taking into account the foundations mentioned above, we can formally state that an application model is a directed clustered graph G = G(A, C), where A is a set of actors and C is a set of communication links. $A_i \subset A$ is an actor. A_i contains input ports $p_{i1}, p_{i2,\ldots}, p_{in} \in P_i$, $P_i \subseteq P$ and output ports $p_{o1}, p_{o2,\ldots}, p_{on} \in P_o$, $P_o \subseteq P$. Actors A_i and A_j exchange data tokens over the communication link $C_{i,j} \in C$, which connects one of the output ports of A_i with one of the input ports of A_j.

Actors can be either atomic (a simple capsule to a sequential computation $a \in A_i$) or composite. Composite actors $W_i \subset W$, $W \subset A$, which are responsible for the hierarchical composition in G, are composed of other actors so that $A_j, A_{k,\ldots}, A_z \subset W_i$.

Opaque composite actors $K_i \subset W$ are composite actors that are able to control their own execution semantics (and that of their non-opaque

Figure 1. Joint validation of application and platform



components). The execution semantics of an opaque composite actor is explicitly defined by a special component called a director d ∈ D, so that ∀ $K_i \subseteq W$, $K_i \supset$ d. A sequencing actor $Q_i \subset Q$ is a special kind of opaque composite actor, and its sole functionality is to guarantee that the incoming data tokens are written to its output ports according to a predefined sequence. This sequence is defined by a UML sequence diagram, which is denoted as a tuple SD = ⟨ L, E, m, n, < ⟩. L is a finite set of lifelines and E a finite set of events. Events can either be send or receive events, s ∈ S, r ∈ R, $S \subset E$, $R \subset E$ respectively. A bijection m: $S \rightarrow R$ matches each send event s to its corresponding receive event r. A surjection n: $E \rightarrow L$ assigns each event e to one and only one lifeline ℓ ∈ L, though a lifeline ℓ can (and it is likely to) be assigned more events. Finally, $< \subseteq$ E × E is an acyclic relation between events consisting of a total order of all e whenever n(e) = ℓ ∈ L, and s < r whenever m(s)=r.

A sequence diagram SD and its respective sequencing actor Q_i are related as follows. A bijection f: $P_i \rightarrow$ S maps each input port p_i of Q_i onto a send event s of SD. Likewise, another bijection b: $P_o \rightarrow$ R maps each output port p_o of Q_i onto a receive event r of SD. The arrival of a data token at an input port p_i can cause the triggering of the send event s = f(p_i) *iff* all preceding events of that lifeline had been triggered previously: ∀e, n(e) = n(s) = ℓ, e < s. Finally, the triggering of the receive event r = b(p_o) can cause the release of the data token through the output port p_o.

Constraint 1: If an actor A_i is connected to a sequencing actor Q_i, all its output ports $p_{o1,}$ $p_{o2,...,}$ $p_{on} \in A_i$ must be connected to input ports $p_{i1,}$ $p_{i2,...,}$ $p_{in} \in Q_i$ so that n(f(p_{i1})) = n(f(p_{i2})) = … = n(f(p_{in})), that is, their respective send events must be assigned to the same lifeline. Likewise, all its input ports $p_{i1,}$ $p_{i2,...,}$ $p_{in} \in A_i$ must be connected to output ports $p_{o1,}$ $p_{o2,...,}$ $p_{on} \in Q_i$ so that n(b(p_{o1})) = n(b(p_{o2})) = … = n(b(p_{on})). Furthermore, n(b(p_{o1})) = n(f(p_{i1})), so that one actor A_i connected to a sequencing actor Q_i can be related to a single lifeline of the sequence diagram governing the behaviour of Q_i. Nothing prevents, however, that a lifeline could be related to multiple actors A_i... A_n.

Figure 1 depicts that each lifeline (L1-L5) of a sequence diagram represents one application actor (denoted by letters a-e) connected to an input or an output port of the sequencing actor. Each message (M1-M6) between lifelines represents communication between two actors of the application model, and the precedence relation (denoted by the vertical dimension on the diagram) defines the order in which messages should be delivered between the

application actors. In fact, the director inside the sequencing actor enforces this order to be maintained.

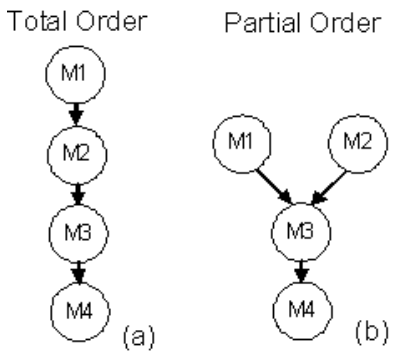
Indrusiak and Glesner (2007) present two different directors for controlling the execution of sequence diagrams, total order and partial order, based on SDTODirector and SDPODirector respectively. The total order director maintains the order of the messages of a sequence diagram whereas the partial order director maintains the order separately on each lifeline. Both of the directors create a precedence graph of all messages of the sequence diagram. Figure 2 depicts the difference between total and partial order precedence graphs. Total order sequence diagram can be illustrated using a directed graph, where the precedence of messages M1-M4 (see the leftmost sequence diagram in Figure 1) is the same as their order in the sequence diagram. Partial order is also a directed graph, but now it is possible to see that the order of messages M1 and M2 does not matter, because the order of messages is enforced separately on each lifeline and M1 and M2 do not have events on the same lifeline.

The directors SDTODirector and SDPO-Director enable the execution of application models within the Ptolemy II framework. In the Application Mapping onto Successively Refined Platform Models Section, we describe the necessary extensions to Ptolemy II so that application models can be mapped onto platform models, and each message of the application is back-annotated with the communication latency of the platform.

PLATFORM MODELLING

From a formal point of view, a platform can be seen as a directed graph H = G(T,V), where T is a set of processors and V are the communication channels through which the processors exchange data packets. However, such models must also contemplate functional details such as internal latencies, congestion effects, and routing and arbitration delays, so that designers can properly explore the NoC design space and optimise the interconnect architecture to a particular traffic scenario (Pande et al., 2005). The use of abstract platform models is a trade-off between the *level of detail* and *model confidence*. The *level of detail* refers to the structural and behavioural abstraction of the NoC's components. The structural abstraction means the granularity of a data storage (that is, a storage for a flit or a packet) and the number of included components and their interconnects. The behavioural abstraction includes how and when the components update their internal state and concurrently interact with other components. The *model confidence* means how useful the model is for a particular purpose for instance in terms of accuracy compared to a reference model.

Figure 2. The difference between total order and partial order precedence graphs



In this article, the reference model is the HERMES infrastructure (Moraes et al., 2004), which is briefly described in the next subsection. Moreover, we present a high-level model based on UML interactions, which derive from the RTL model of HERMES infrastructure called RENATO. Our next model, JOSELITO, is a simplified NoC model that uses the Payload Abstraction Technique (PAT) and allows performance evaluation by combining simulation and analytical methods. Furthermore, BOÇA is another simplified model abstracting all the internal state of a router except its buffers. Finally, we give an overview of the actors developed for observing and debugging the execution of an application running on top of the NoC models, called scopes.

HERMES Reference Model

HERMES is a Register Transfer (RT) level infrastructure, which implements a low area overhead packet switching NoC. HERMES supports both 2D mesh and torus topologies and allows designers to select routing algorithm, control flow mechanism, flit size, and buffer depth. Its routers have centralised switching control logic and five bi-directional ports. One port is used to establish the communication between a router and its local processing element, whereas the others are connected to the neighbour routers. HERMES uses round-robin arbitration for granting access to incoming packets, which are stored in a FIFO buffer. The priority of an input port is a function of the last input port having a routing request granted. If the incoming packet request is granted by the arbiter, the XY routing algorithm is executed to connect the input port to the correct output port. If the algorithm returns a busy output port, the header flit and all subsequent flits of this packet are blocked. After all flits in a packet are transmitted, the port is released (Moraes et al., 2004).

RENATO

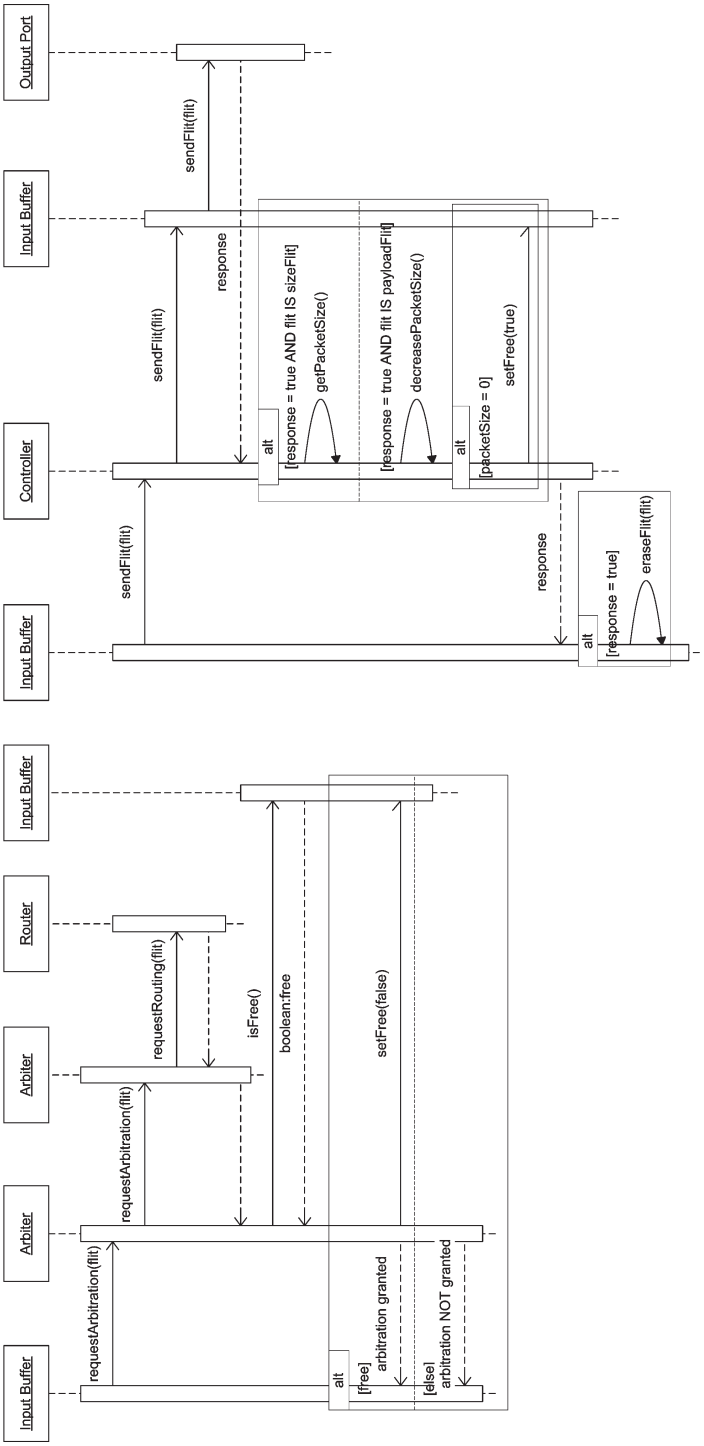
RENATO is an actor-oriented model based on two UML sequence diagrams, which describe the interactions of the HERMES router components. These UML sequence diagrams, shown in Figure 3, are the arbitration request by a particular input buffer (interaction on the left side of the figure) and the transmission of a flit from one input buffer to a neighbour router through an output channel (interaction on the right side of the figure) (Indrusiak et al., 2008).

The interaction on the left side of Figure 3 illustrates the establishment of a connection between an input buffer to an output port. Initially, the input buffer sends the packet header to the routing controller. The routing controller asks the arbiter to choose one of the possible incoming requests that can arrive from any of the five input buffers. After getting a positive response, the arbiter sends the header flit to the router and the router executes a particular algorithm (for instance XY algorithm) to determine which output port the packet should be sent to. Once the routing is done, the controller verifies if the chosen output port is free. If the output port is free, the input buffer establishes the connection to the output port, otherwise the connection is refused and the input buffer must start the whole input-output connection requesting process again.

The interaction on the right side of Figure 3 models the transmission of a flit between two neighbouring routers. The controller receives the flit from the local input buffer and checks which output port is allocated to it. After sending the flit, the controller waits for an acknowledgement. A positive acknowledgement removes the successfully sent flit from the source input buffer, a negative acknowledgement causes a re-sending of the flit.

Neither of the sequence diagrams in Figure 3 considers timing among the interactions. In order to improve the accuracy of RENATO, timing information extracted from the simulation

Figure 3. UML sequence diagrams depicting interactions between components of a HERMES NoC



of the HERMES VHDL model was added to the respective transaction model. When validating the model, performance results of RENATO were compared to the results of HERMES (cycle-accurate simulation) using different traffic scenarios and NoC configurations (Indrusiak et al., 2008). For long-lasting traffic, the error is only about 10%, which is a very good figure considering that the actor-oriented model is based on the interactions only and works without a synchronising clock signal.

JOSELITO

JOSELITO is a more abstract platform model than RENATO. However, it uses the same UML interactions defined in Figure 3. The main difference between JOSELITO and RENATO is the decrease in JOSELITO's simulation time, caused by the reduction of the number of communication events due to the flit by flit packet forwarding. JOSELITO uses the Payload Abstraction Technique (PAT), which comprises that: *(i)* the packet is defined as a *header* and a *trailer, (ii)* the buffer is a FIFO structure modelled as a finite state machine, *(iii)* packet headers are released from a given router once there is available buffer space at next hop on its route, and *(iv)* a simple analytical method is used to calculate the packet trailer release time (*ptrt*) (Ost et al., 2008).

Even if the transmission of the packet payload is abstracted away, the simulation model can still represent both unblocked and blocked packet transmission scenarios. If no resource conflicts occur (unblocked scenario), the latency and throughput of the NoC can be measured with no loss of accuracy. In a blocked scenario, when a header packet arrives in an input buffer, two blocking situations can occur: either the desired output port is reserved to another input port or the target neighbour input buffer is not able to receive a header or a trailer of the packet. In such cases, a connection of an intermediate router (for instance, a router between the source and the target processing elements) can be closed without considering the impact of the blockage to other packets.

In this context, the *ptrt* is used to ensure a correct functionality during packet transfers, allowing the designer to obtain high accuracy latency and throughput results with shorter simulation time in comparison with RENATO. The accuracy of those results also depends on the parameters obtained from the RTL simulation. One possible alternative for the analytical calculation of the *ptrt*, presented by Ost et al. (2008) and adopted within the current work, is shown in Equation (1).

$$ptrt = hft + pcksize * ctf \quad (1)$$

where *ptrt* is the packet trailer release time,

hft is the header forwarding time,

pcksize is the packet size (number of flits),

and c*tf* is the number of clock cycles to transmit one flit

The results presented by Ost et al. (2008) show that the worst case latency of JOSELITO is 5.26% lower than the latency of HERMES. Furthermore, JOSELITO was in average 2.3 times faster than RENATO in 88% of the executed cases. It is important to mention that the analytical method is not restricted to the three parameters presented in Equation (1). Therefore, different parameters (such as buffer capacity) can be included in order to improve the *ptrt* estimation.

BOÇA

The third platform model considered in this article is the fastest and most abstract one, but it obviously pays for that by having the lowest accuracy. BOÇA considers the multi-hop nature of NoC communications and the buffering of flits at the input ports of each router, but it abstracts completely the arbitration logic and internal state of the routers. Therefore, BOÇA's way to handle the interference among traffic flows does not reflect a real implementation.

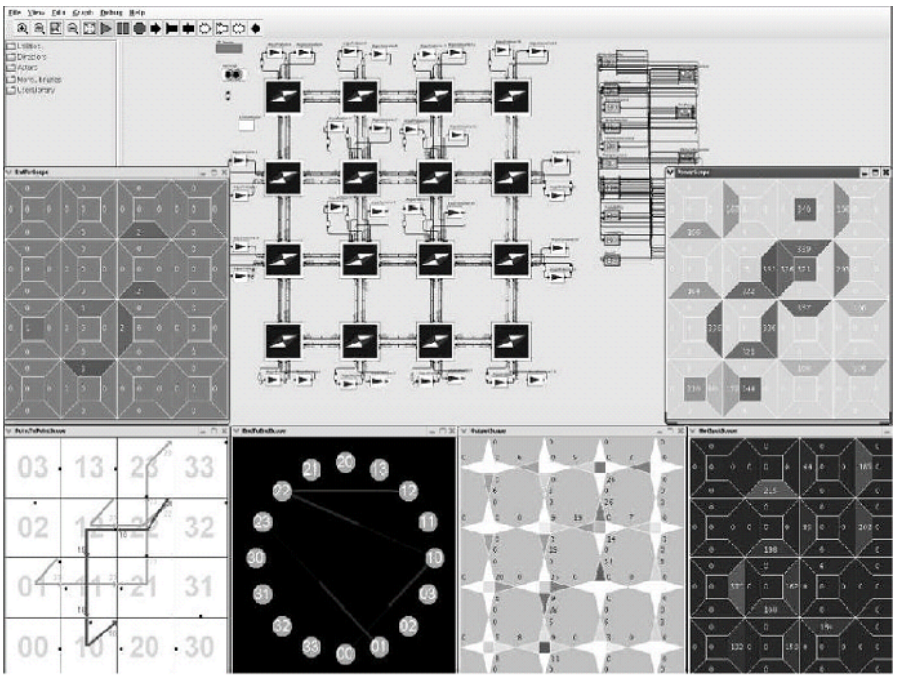
Debugging Support

In order to increase the observability and debugging capabilities of the platform models, we have created actors to monitor traffic, collect data from the network, and display it graphically. All those extensions are combined to a small framework called *NoCScope*. The *NoCScope* is a composition of seven scopes: *BufferScope*, *InputScope*, *OutputScope*, *PowerScope*, *HotspotScope*, *EndToEndScope*, and *PointToPointScope*. Figure 4 presents a screen capture of the model running with six of the scopes (*PointToPointScope*, *EndToEndScope*, *OutputScope*, and *HotSpotScope* in the bottom of the picture and *Bufferscope* and *Powerscope* above them). The *Bufferscope* is currently used to analyse buffer occupation, from which we can determine the router memory requirements. The *InputScope* and the *OutputScope* capture the activity of each input and output channel of the routers. With the *PowerScope* we obtain the power consumption of the routers by analysing the transition activity on the channel, as Palma et al. (2006) have done. The *HotspotScope* emphasizes and quantifies the output ports of the NoC routers that are trying to send packets to the next router, but are being blocked due to congestion. The *EndToEndScope* gives us an overview of which modules are communicating with each other and the *PointToPointScope* presents the complete path that the packets use inside the NoC. In all cases, the displays are updated as the simulation runs. Therefore, we can have full observability and controllability of the dynamic behaviour of the NoC model.

APPLICATION MAPPING ONTO SUCCESSIVELY REFINED PLATFORM MODELS

The major goal of the work this article presents is to guide system designers to choose and parameterise a platform model that can cope

Figure 4. Application model, platform model, and the scopes



with the requirements imposed by a particular application. In previous sections, we have described the proposed approaches for modelling the application and platform. In this section, we present a methodology that allows the joint validation of such models, refining the idea that Määttä et al. (2008) present, but considering multiple platform models that range from more abstract and fast ones to more accurate ones. We propose a methodology based on successive refinement of platform models, so that designers can explore the trade-off between the platform model accuracy, observability, and execution speed. Concretely this means that a designer should start by validating the application on top of a fast but not very accurate model of the platform, and refine such model step-wise so that the accuracy is increased at the expense of the model's execution speed. This enables speeding up the design space exploration.

Conceptually, the link between the application and platform models is established so that a particular application task should be executed by a particular processor of the platform model. This process is known as application-platform mapping and has received substantial attention by the research community within the past few years (Hu & Marculescu, 2005; Marcon et al., 2005; Murali et al., 2005). Thus, it is necessary to define the mapping in order to validate the performance of an application executed on a particular platform.

Following the formal definitions the Application Modelling and Platform Modelling Sections have presented, we define mapping as a function u: $L \rightarrow T$, which defines that a lifeline $\ell \in L$ is assigned to a processor $t \in T$. By assigning a lifeline ℓ to a processor t, the mapping process denotes that all the sequential computations $a_1, a_2, \ldots, a_n$ that are encapsulated by the actors related to the lifeline ℓ are executed by the processor t. Again, we mean here that an actor is related to a lifeline *iff* its ports are connected to the ports that are mapped to the events contained by that lifeline, as described in Constraint 1.

In order to have an implementation of such a mapping function, we introduce an additional element to our approach, the mapper, which has access to information from both the application and platform models, and reacts to events that the directors of sequencing actors and the abstract processing elements generate. The mapper assigns each of the lifelines (lifelines L1 to L5 in Figure 1) to one abstract processing element at the platform model (denoted as PE in Figure 1).

For each token a sequencing actor receives, a pre-defined message within its sequence diagram will be triggered (for example, the message M1 sent by lifeline L1 to lifeline L2). When this happens, the corresponding director D2 interrupts the delivery and notifies the mapper about the message. Since the mapper is responsible of assigning each lifeline to an abstract processing element, it knows that for instance lifeline L1 is mapped to PE 2, whereas L2 is mapped to PE 7. Once the mapper receives the information about the triggered message, it will command the processing element associated to the sender of the message (PE 2) to generate the corresponding traffic into the interconnect structure (in the case of a NoC platform, it must create a packet with destination, size, and payload, and write this packet on the local input port of the corresponding router). Then the mapper waits until the processing element associated to the receiver of the message (PE 7) notifies the complete reception of the packet. Upon notification, the mapper calls back the director D2 (which has notified the triggering of the message) and informs it that the message can now be delivered. After that, the director can forward the message to the output port of the sequencing actor, and the message reaches its destination with the exact latency that it would take if the application is executed on top of the implementation platform.

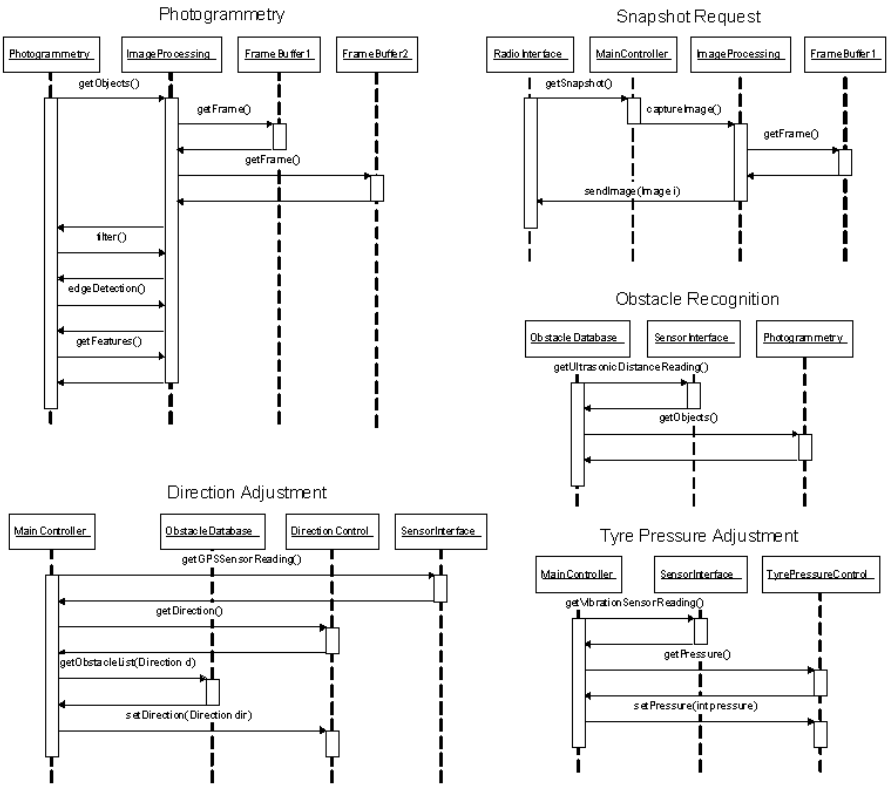
As the mapper implementation has a critical influence on the communication costs, our approach considers the choice of the mapper as a design decision. Therefore, our implemented approach is extensible, so that we can create a library of different mapping algorithms and heuristics. However, this is not within the scope of this article. The current mapper implemen-

tation supports only static mapping and has no constraints (each application actor can be mapped to any processing element). These are not restrictions of the modelling approach, and future work will explore the possibility of different static and dynamic mapping heuristics, as well as the sharing of processing elements by multiple application actors.

CASE STUDY

Figure 5 illustrates the sequence diagrams of the application model we have used in this case study. The application is an autonomous vehicle that is modelled in five different sequence diagrams: photogrammetry, obstacle recognition, direction adjustment, tyre pressure adjustment, and snapshot request. The vehicle has two cameras, which capture images from the direction the vehicle is moving. The photogrammetry logic of the system pre-processes the images and extracts three-dimensional features by exploring the stereoscopy in both images. The obstacle recognition extracts the coordinates of the possible obstacles that might force the vehicle to adjust its direction. The obstacle coordinates are then fed to the obstacle database, which is also adjusted with the information from the ultrasonic sensor (which can measure more precisely the distance of obstacles that are closer). The direction adjustment logic can determine the vehicle's current position using GPS and then adjust the direction according to the information of the obstacle database in order to avoid collisions with obstacles. The vehicle contains also sensors able to measure vibration.

Figure 5. UML sequence diagrams of an autonomous vehicle application



If the vehicle vibrates too much, it affects the quality of the camera images. Therefore, the tyre pressure adjustment is able to adjust the vehicle's tyre pressure so that it is suitable for the surfaces it moves on. Finally, a radio interface enables interaction with external entities. In this example, the radio interface is only used to command the capture of images.

We have simulated the application on three different platform models, RENATO, JOSELITO, and BOÇA, all of them implementing a 4x4 mesh topology. The packets that the application actors send to each other have a maximum size of 48 flits, each flit of 16 bits. Empirically this packet size seems to be a good trade-off between the overhead the multiple packet headers generate versus the long term occupation of platform resources, such as channels and buffers. Application messages that are larger than 48 * 16 bits are divided in multiple packets.

We have used two different random mappings for each platform and have measured the network latency for each message of each sequence diagram. We have measured the network latency from the point the processing element sends a packet containing the message to the point the packet arrives at the processing element of the target node. That is, how long a packet spends in the network, either routed between switches or in an input buffer of a switch waiting for routing. Therefore, the latency of messages depends on the network congestion, data size of the corresponding packet, and the mapping (that is, if the sending and receiving actors of the packet are mapped to nodes that are close to each other or not).

We have fixed the operation frequency of the platforms to 50 MHz and simulated the system for 18 seconds of wall-clock time. Application-specific constraints define the execution frequency of each sequence diagram. The photogrammetry, obstacle recognition, and direction adjustment are executed once every two seconds, while the tyre pressure adjustment and snapshot request are executed once a second.

Table I presents the worst case latency results for two different mappings of the application over RENATO, JOSELITO, and BOÇA. Since RENATO is the closest model to a RTL implementation, it is used as a reference within this analysis. As expected, BOÇA presents a significant error for the worst case latency (around 45% in comparison with RENATO). This is due to the fact that it is not back-annotated with timing delays from a real implementation and its modelling of network arbitration is very simplistic. However, when simulating the application model in the PtolemyII environment, simulating the application over BOÇA was 402 times faster for mapping 1 and 492 times faster for mapping 2 in comparison with RENATO.

Table 1. Worst case latency results of two different mappings of the same application mapped onto the platform models (latency figures in milliseconds, error percentage compared to the reference model)

	Mapping 1			**Mapping 2**		
	RENATO	JOSELITO	BOÇA	RENATO	JOSELITO	BOÇA
Direction Adjustment	0,39	0,27 (30%)	0,24 (37%)	0,38	0,26 (30%)	0,25 (32%)
Obstacle Recognition	0,12	0,09 (29%)	0,07 (41%)	0,10	0,07 (29%)	0,07 (29%)
Photogrammetry	1,41	0,99 (30%)	0,54 (62%)	1,37	0,95 (31%)	0,52 (61%)
Snapshot Request	1,35	0,94 (30%)	0,90 (33%)	1,31	0,90 (31%)	0,89 (32%)
Tyre Pressure Adjustment	0,14	0,09 (32%)	0,01 (55%)	0,11	0,08 (30%)	0,05 (50%)
Total	3,41	2,38 (30%)	1,76 (46%)	3,27	2,26 (31%)	1,78 (45%)

As soon as we have executed our application on BOÇA, we have used JOSELITO to extract more information about which mapping would work better when considering a mesh topology network. Even though JOSELITO has a high worst case latency error (around 30%), JOSELITO was proved to be useful, because every time JOSELITO has lower latency, RENATO has lower latency as well. The simulation of the application on JOSELITO was 2.8 times faster for mapping 1 and three times faster for mapping 2 in comparison with RENATO.

One additional observation we can make from Table I is related to the long worst case latency for photogrammetry and snapshot request sequence diagrams. These long latencies are due to the large size of some of their messages. Only those two sequence diagrams are involved in transferring image frames from one processing element to another.

An important issue when modelling systems in different abstraction levels is the relative order of the platform models when exploring the design space of different features, such as mapping. Table I shows the relative ranking of RENATO, JOSELITO, and BOÇA for the two mappings. The same order is observed in both mappings. These results point out the fidelity among different platform models, enabling design space exploration at higher abstraction levels.

CONCLUSION AND FUTURE WORK

This article has presented an application modelling formalism that supports the joint validation of application and platform models. This approach is based on executable models, and uses a back-annotation strategy to increase the accuracy of the execution of a given application by considering the timing behaviour obtained from the platform on which the application runs.

In order to support a top-down design flow, the proposed formalism supports the successive refinement of platform models at multiple abstraction levels. Fast and abstract platform models should be used early on the design flow in order to perform rough evaluations and to rule out poorly performing platforms. Accurate models should be used later for fine-tuning platform parameters and choosing the best mapping.

We have presented three NoC models, each of them having different strengths regarding simulation speed, accuracy, observability, and modelling effort. We have simulated two different mappings of the same application on all three platform models, and therefore found out that the trends of latency are consistent across abstraction levels. The formalised definition of application, platform, and mapping proposed in this article allows a plug-and-play approach what comes to jointly validate an application mapped onto a platform. The swapping of platforms is effortless, and a given application can be simulated over different platforms by simply choosing each platform template (and mapping heuristic) from a library.

In the future, we will explore the effect of different static and dynamic mapping heuristics on the communication latency of an application. We will also consider the effect of mapping several application actors onto the same processing element in order to reduce the network latency (at the expense of increased computation times due to the multitasking overhead). Furthermore, we intend to support UML 2.0 sequence diagrams including combined fragments, so that designers can specify applications with communication behaviour that includes parallel, optional, and iterative triggering of messages.

REFERENCES

Agha, G. A. (1986). *ACTORS: A model of concurrent computation in distributed systems*. Cambridge, MA: MIT Press.

Arpinen, T., Kukkala, P., Salminen, E., Hännikäinen, M., & Hämäläinen, T. D. (2006). Configurable multiprocessor platform with RTOS for distributed execution of UML 2.0 designed applications. In *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition* (pp. 1324-1329).

Brooks, C., Lee, E. A., Liu, X., Neuendorffer, S., Zhao, Y., & Zheng, H. (2007). *Heterogeneous concurrent modeling and design in Java: Volume 1: Introduction to Ptolemy II*. Berkeley, CA: EECS Department, University of California.

Eker, J., Janneck, J. W., Lee, E. A., Liu, J., Liu, X., & Ludvig, J. (2003). Taming heterogeneity - the Ptolemy approach. *Proceedings of the IEEE*, *91*(1), 127–144. doi:10.1109/JPROC.2002.805829

Grötker, T., Liao, S., Martin, G., & Swan, S. (2002). *System design with SystemC*. Norwell, MA: Kluwer Academic Publishers.

Hewitt, C., Bishop, P., & Steiger, R. (1973). A universal modular actor formalism for artificial intelligence. In . *Proceedings of the IJCAI*, *III*, 235–245.

Hu, J., & Marculescu, R. (2005). Energy- and performance-aware mapping for regular NoC architectures. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, *24*(4), 551–562. doi:10.1109/TCAD.2005.844106

Indrusiak, L. S., & Glesner, M. (2007). Specification of alternative execution semantics of UML sequence diagrams within actor-oriented models. In *Proceedings of the Symposium on Integrated Circuits and System Design* (pp. 330-335).

Indrusiak, L. S., Ost, L., Möller, L., Moraes, F., & Glesner, M. (2008). Applying UML interactions and actor-oriented simulation to the design space exploration of network-on-chip interconnects. In *Proceedings of the IEEE Computer Society Annual Symposium on VLSI* (pp. 491-494).

Indrusiak, L. S., Thuy, A., & Glesner, M. (2007). Executable system-level specification models containing UML-based behavioral patterns. In *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition* (pp. 301-306).

Kempf, T., Karuri, K., Wallentowitz, S., Ascheid, G., Leupers, R., & Meyr, H. (2006). A SW performance estimation framework for early system-level-design using fine-grained instrumentation. In *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition* (pp. 468-473).

Lei, T., & Kumar, S. (2003). A two-step genetic algorithm for mapping task graphs to a network on chip architecture. In *Proceedings of the Euromicro Symposium on Digital Systems Design* (pp. 180-189).

Määttä, S., Indrusiak, L. S., Ost, L., Möller, L., Nurmi, J., Glesner, M., et al. (2008). Validation of executable application models mapped onto network-on-chip platforms. In *Proceedings of the IEEE Symposium on Industrial Embedded Systems* (pp. 118-125).

Marcon, C., Borin, A., Susin, A., Carro, L., & Wagner, F. (2005). Time and energy efficient mapping of embedded applications onto NoCs. In *Proceedings of the Asia South Pacific Design Automation Conference* (pp. 33-38).

Martin, G. (2002). UML for embedded systems specification and design: Motivation and overview. In *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition* (pp. 773-775).

Moraes, F., Calazans, N., Mello, A., Möller, L., & Ost, L. (2004). Hermes: An infrastructure for low area overhead packet-switching networks on chip. In *Integration, the VLSI Journal, 38*(1), 69-93.

Murali, S., Benini, L., & De Micheli, G. (2005). Mapping and physical planning of networks-on-chip architectures with quality-of-service guarantees. In *Proceedings of the Asia South Pacific Design Automation Conference* (pp. 27-32).

Object Management Group. (2005). *UML profile for schedulability, performance, and time specification, version 1.1*. Retrieved April 27, 2009, from http://www.omg.org/docs/formal/05-01-02.pdf

Ost, L., Möller, L., Indrusiak, L. S., Moraes, F., Määttä, S., Nurmi, J., et al. (2008). A simplified executable model to evaluate latency and throughput of networks-on-chip. In *Proceedings of the Symposium on Integrated Circuits and Systems Design* (pp. 170-175).

Palma, J. C. S., Indrusiak, L. S., Moraes, F. G., Ortiz, A. G., Glesner, M., & Reis, R. (2006). Adaptive coding in networks-on-chip: Transition activity reduction versus power overhead of the Codec Circuitry. In Proceedings of *PATMOS* (pp. 603-613).

Pande, P. P., Grecu, C., Jones, M., Ivanov, A., & Saleh, R. (2005). Performance evaluation and design trade-offs for network-on-chip interconnect architectures. *IEEE Computer*, *54*(8), 1025–1040. doi:10.1109/TC.2005.134

Patel, H. D., & Shukla, S. K. (2004). *SystemC kernel extension for heterogeneous system modeling – a framework for multi-MoC modeling & simulation*. Norwell, MA: Kluwer Academic Publishers.

Riccobene, E., Scandurra, P., Rosti, A., & Bocchio, S. (2005). A SoC design methodology involving a UML 2.0 profile for SystemC. In *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition* (pp. 704-709).

Ristau, B., Limberg, T., & Fettweis, G. (2008). A mapping framework for guided design space Exploration of heterogeneous MP-SoCs. In *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition* (pp. 780-783).

Saint-Jean, N., Benoit, P., Sassatelli, G., Torres, L., & Robert, M. (2007). Application case studies on HS-scale, a MP-SOC for embbeded systems. In *Proceedings of the International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation (IC-SAMOS)* (pp. 88-95).

Sanna Määttä received her MSc degree in information technology from Tampere University of Technology, Finland, in 2004. Currently she is a PhD student at the Department of computer systems at Tampere University of Technology. She works there since 2003, first as a research assistant and after graduation at 2004 as a researcher. In 2007-2008 she has worked one year as a guest researcher at the Institute of Microelectronic Systems at the Technical University of Darmstadt, Germany. Currently she is working in a SYSMODEL project and her main research interests are high level modelling and validation of applications, networks-on-chip, and multiprocessor systems.

Leandro Möller is currently a PhD student at the Technical University of Darmstadt, Germany. He is a member of the Microelectronic Systems group in this same University. He received his master and bachelor degree in computer science, in 2005 and 2003, respectively, both from the Catholic University of Rio Grande do Sul, Brazil. Since 2001 he is also a member of the Hardware Design Support Group (GAPH) at Catholic University of Rio Grande do Sul. Further he received two awards, the first place in the Xilinx Student Contest given during the SBCCI conference in 2005 and the Europractice Design Contest Award in the conceptual design category in 2006 given during the DATE conference. In the summer of 2006, he was an intern at Xilinx Research Labs, San Jose, US. His research interests are partial and dynamical reconfiguration of FPGAs, Networks-on-Chip and High Level Modeling of Systems-on-Chip.

Leandro Soares Indrusiak received an Eng. degree in electrical engineering from Federal University of Santa Maria (UFSM), Brazil, in 1995, a MSc degree in computer science from Federal University of Rio Grande do Sul (UFRGS), Porto Alegre, Brazil, in 1998, and a doctoral degree in computer science jointly issued by Technische Universitaet Darmstadt, Germany, and the Federal University of Rio Grande do Sul, in 2003. He is a lecturer in real-time embedded systems at the Computer Science department of the University of York, United Kingdom. Previously he was a research fellow at the Microelectronic Systems Institute of the Technische Universitaet Darmstadt and an assistant professor at the Catholic University of Rio Grande do Sul, Brazil. His research interests include embedded systems, hardware/software design automation, distributed and concurrent computing, system modeling and simulation. Luciano Copello Ost received bachelor degree and a MSc degree in computer science from the Catholic University of Rio Grande do Sul, Brazil, respectively, in 2001 and 2004. From 2004 to 2006 he worked as Research Assistant at PUCRS.

Luciano Ost is currently a PhD student at PUCRS and a member of the Hardware Design Support Group (GAPH) in this same University. Further, he worked one year as invited research at the Microelectronic Systems Institute of the Technische Universitaet Darmstadt. His main research interests are digital systems design, NoCs (networks on chip), and high level modeling of MPSoCs (multi-processor systems on chip). Dr. h. c. mult.

Manfred Glesner was born in Saarlouis/Germany and got his university education at the Saarland University in Saarbrücken/Germany. He studied applied physics and electrical engineering and received his diploma in 1969. In 1975 he received the PhD degree from the same university for his research work on optimisation techniques in computer aided circuit design. In 1981 he became assistant professor in the Faculty for Electrical Engineering and Information Technology of Darmstadt University of Technology. Since 1989 he owns the chair for Microelectronic Systems at Darmstadt University. For his contributions to the field of microelectronics he has received three honorary causa doctoral degrees, one from Tallinn Technical University (Estonia), one from Bucharest Polytechnical University (Romania), and one from Mongolian Technical University in Ulaan Bator (Mongolia). In 2008, he received the decoration Palmes académiques in the grade of Chevalier.

Jari Nurmi is professor of computer systems at Tampere University of Technology (TUT) since 1999. He has had research, education, and management positions at TUT and industry 1987-1998. He received his PhD degree from TUT in 1994. His current research interests include system-on-chip, network-on-chip, embedded (multi-)processor architectures, and implementations of communication, positioning and DSP systems. He is leading a group of about 25 researchers. Dr. Nurmi is the general chairman of the international symposium on System-on-Chip (SoC) 1999-2026, board member of SoC, FPL, and NORCHIP. He was the general chair of FPL'05 and SiPS'09 conferences. He was heading TELESOC graduate school 2001-2005. He has (co-) authored over 180 international papers, edited Processor Design: System-on-Chip Computing for ASICs and FPGAs (Springer), and co-edited Interconnect-centric Design for Advanced SoC and NoC (Kluwer). He is also an associate editor of IJERTCS. He has supervised over 100 MSc and 10 Doctoral theses, is a senior member in IEEE, has received Nokia Educational Award 2004, and Tampere Congress Award 2005. He was also Academy of Finland Research Fellow 2007-2008.

Fernando Moraes received the electrical engineering and MSc degrees from the Universidade Federal do Rio Grande do Sul (UFRGS), Porto Alegre, Brazil, in 1987 and 1990, respectively. In 1994 he received the PhD degree from the Laboratoire d´Informatique, Robotique et Microélectronique de Montpellier (LIRMM), France. He is currently at PUCRS, where he has been an associate professor from 1996 to 2002, and professor since 2002. He has authored and co-authored twelve peer refereed journal articles in the field of VLSI design, comprising the development of networks on chip and telecommunication circuits. He has also authored and co-authored more than 120 conference papers on these topics. He has co-advised 3 MSc, advised 13 MsC, co-advised 3 PhD and advised 1 PhD works. Now he is responsible for the computer science graduate program at PUCRS. His research interests include microelectronics, FPGAs, reconfigurable architectures, and NoCs (networks on chip).