Applying UML Interactions and Actor-oriented Simulation to the Design Space Exploration of Network-on-Chip Interconnects

Leandro Soares Indrusiak[1], Luciano Ost[1,2], Leandro Möller[1,Ψ], Fernando Moraes[2,Ψ], Manfred Glesner[1]

[1] FG MES - Technische Universität Darmstadt
Karlstr. 15, 64283 Darmstadt, Germany
[indrusiak | moller | glesner]@mes.tu-darmstadt.de

[2] Catholic University of Rio Grande do Sul (PUCRS)
Ipiranga, 6681 - 90619-900 - Porto Alegre, Brazil
[ost | moraes]@inf.pucrs.br

Abstract

This paper presents an approach supporting designer-driven interactive design space exploration for Network-on-Chip interconnects. It abstracts the functionality of the interconnect using UML interactions, which are in turn used as reference for the development of an actor-oriented model. Such model can be annotated with timing information, thus allowing the validation of the interconnect performance under a given traffic load. The proposed model allows simpler tuning and modification of the interconnect, improved observability and debugging, while presenting acceptable loss of accuracy with regard to a cycle-accurate RTL model.

1. Introduction

The design space of packet-switching on-chip interconnects such as Networks-on-Chip (NoCs) is very wide. Designers can explore a wide range of routing algorithms, topologies, buffering and flow control schemes, and each of such choices will affect how efficiently a particular NoC handles the traffic generated by a given multiprocessing application (of family of applications). This paper addresses the difficulties met by a designer when trying to interactively navigate through such a wide design space. Such a scenario includes the system-level specification of an initial solution, the evaluation of design alternatives through formal methods and/or simulation and the refinement of the specification towards a reference design that can be used for hardware synthesis.

In practice, such ideal top-down scenario is not always possible. Most NoC models reuse previously designed modules such as arbiters or FIFO buffers, and such modules are usually implemented at RTL (Register Transfer Level) or logic levels using HDLs. Thus, even if parts of the interconnect architecture are designed or specified using more abstract models, the only reference model that describes the complete functionality of the interconnect is usually the one implemented using HDL. This makes the interactive design space navigation and the exploration of design alternatives more difficult, since HDL models simulate slowly, are hard to change and require additional expertise if they are to be validated with complex testbenches.

A common practice to facilitate the exploration of design alternatives is to build a simplified model that abstracts irrelevant parts of the RTL description of the NoC, allows for quick "what-if" experiments and provides a simpler interface to verify its performance and functionality. However, the usefulness of such abstract models depends on how well they can capture the original behavior described at RTL. The abstract modeling activity becomes a trade-off between precision (which is associated to complexity and slow simulation speeds) and abstraction (which, if excessive, renders the model useless).

This paper reports a number of attempts to create an abstract model of a well known NoC architecture and focuses on the difficulties to find a proper abstraction of RTL modeling constructs. The development of such abstract models must contemplate functional details such as internal latencies, congestion effects, routing and arbitration delays, so that designers can properly explore the NoC design space, optimizing the interconnect architecture to a particular traffic scenario. Figure 1 depicts the approach used in this paper, highlighting the fact that the overhead due to the abstract modeling can be compensated by the potential improvement on the final design due to the extensive analysis and exploration of alternatives at a higher level of abstraction.

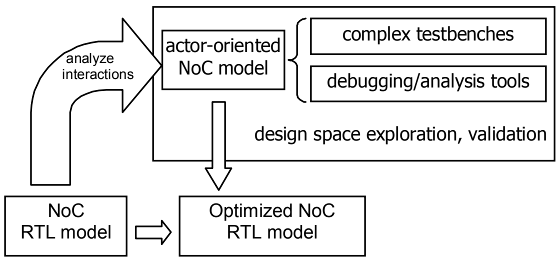


Fig. 1 – Proposed approach for NoC modeling and design space exploration.

2. Network-on-Chip architecture

The HERMES NoC [1] was chosen to be used within this work due to its public domain distribution and its simple yet powerful architecture. HERMES is a parameterizable infrastructure specified in VHDL at RT level, aiming to implement low area overhead packet switching NoCs. In such interconnects, all data is partitioned in packets, which are composed by a header and a number of flits (data words). Packets are sent from a

Ψ The authors acknowledge the support of CNPq - Brasil.

processing element to a given destination through routers that have some awareness of the interconnect topology and thus manage to deliver the packet while avoiding deadlocks and livelocks.

HERMES supports either 2D mesh or torus topologies and allows designers to select routing algorithm, control flow mechanism, flit size and buffer depths. Its routers have centralized switching control logic and five bi-directional ports. One port is used to establish the communication between a router and its local processing element, while the others are connected to the neighbor routers. Each input port stores received data on a FIFO buffer. The routing algorithm chosen for this work is the XY, which sends packets in the X direction up to the target X coordinate, and next proceeds in the Y direction until reaching the target router. Multiple packets may arrive simultaneously in a given router. A centralized round-robin arbitration grants access to incoming packets. 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.

3. Abstract modeling

Two major aspects must be addressed when supporting the creation of NoC interconnect models at higher level of abstraction: structural abstractions and behavioral abstractions. Structural abstractions include how the NoC subsystems are divided, through which channels they communicate and which kind of data they exchange. On the other hand, the behavioral abstractions describe how and, more importantly, when such subsystems update they internal state and concurrently interact with other subsystems.

3.1. Actor-oriented modeling in Ptolemy II

Actor orientation design is a methodology that separates functionality concerns (represented specifically by actors) from the component interaction concerns (formalized as concurrent models of computation) [2][3]. Thus, the communication among actors is based on the exchange of data tokens through channels, and is managed by a director that follows a given model of computation (MoC). The MoC defines how concurrency and time affect the way actors communicate and behave. Ptolemy II is a graphical modeling and design environment implementing such an actor-oriented design methodology. Examples of MoCs implemented in Ptolemy II include continuous time, discrete-event and synchronous dataflow.

3.2. Interaction-based analysis

To better understand and abstract the relevant behavioral patterns in the RTL model of the NoC interconnect, each inter-component interaction was formalized using UML sequence diagrams [4]. Such approach was required after a number of failed attempts to simply convert the VHDL code into a more abstract model. In such attempts, the numerous implementation-related details present in the VHDL model were often preventing the proper capture of the conceptual behavior of the system.

In this work, interactions describe all inter-component transactions occurring since the arrival of a flit on the input buffer of a given NoC router until its output to the input buffers of a neighbor router. Two distinct UML sequence diagrams, shown in Fig. 2, depict those interactions: (a) the arbitration request by a particular input buffer and (b) the transmission of a flit from one input buffer to a neighbor router through an output.

Interaction (a) illustrates an arbitration request by an input buffer that has previously received a new packet. It sends the packet header flit to the router controller as a request and waits for a response (as denoted by the usage of the UML notation for a synchronous message). Many instances of this transaction may occur concurrently, as many input buffers may request arbitration. The controller asks the arbiter to choose one of the incoming requests, which is performed according to a particular algorithm such as round-robin. The controller then reads the incoming header flit from the chosen input buffer and discards all others. It extracts from that header flit the information about the packet destination and sends it to the router. The router then follows a particular algorithm to determine to which output direction the incoming packet should be sent. The direction information is returned to the controller, which in turn checks if the output port to that direction is free. If the output port is free, the controller allocates it to the input buffer, otherwise it doesn't.

Interaction (b) models the transmission of a flit from an input buffer to the input buffer of a neighbor router. The controller receives the flit from the local input buffer and checks which output port is allocated to it (the port allocation was done previously in interaction (a); in case there is no allocated output port, interaction (a) is performed instead). It then sends the flit to the remote input buffer through that port, and waits for an receipt acknowledge. The controller is configured to keep track of the different parts of a data packet, so it will read the packet size (if it is the second flit of a packet in the case of HERMES) or update its counter of sent flits (for all subsequent payload flits) upon receiving the acknowledge from the remote input buffer. Finally, the controller notifies the local input buffer about the success of the flit transfer. If not successful, the input buffer will then repeat the whole interaction.

It is worth noticing that the sequence diagrams simply define a partial order between messages, without any particular reference to time and exposing potential concurrency. Such untimed model is enough to analyze the functionality of the interconnect, but it can't provide an accurate behavior of the system over time. For instance, it is not possible to determine the latency of a given packet or the occupation of a given input buffer. To obtain accurate timing behavior, which in turn can provide metrics for proper design space exploration, this work annotates timing information for each transaction depicted on an UML interaction. Such timing information can be obtained from a cycle-accurate RTL simulation or can be a design-time estimate. In both cases, such models become a versatile tool for design space exploration, since it allows a simple way to explore design alternatives. For example, the message *sendFlit* between the output port and the remote input buffer in interaction (b) can have alternative implementations using a handshake protocol or a credit-based flow control. By annotating the message with timing information from a cycle-accurate simulation, a designer can compare the impact of the use of each one of them (in the case of HERMES, a handshake takes twice as much time as a credit-based flow control). The basic architecture of HERMES is also used by many similar NoCs [5], but they all exhibit different timing behavior. Thus, another advantage of the proposed approach is the fact that a designer can explore alternative NoCs simply by changing the annotations within the identified interactions.

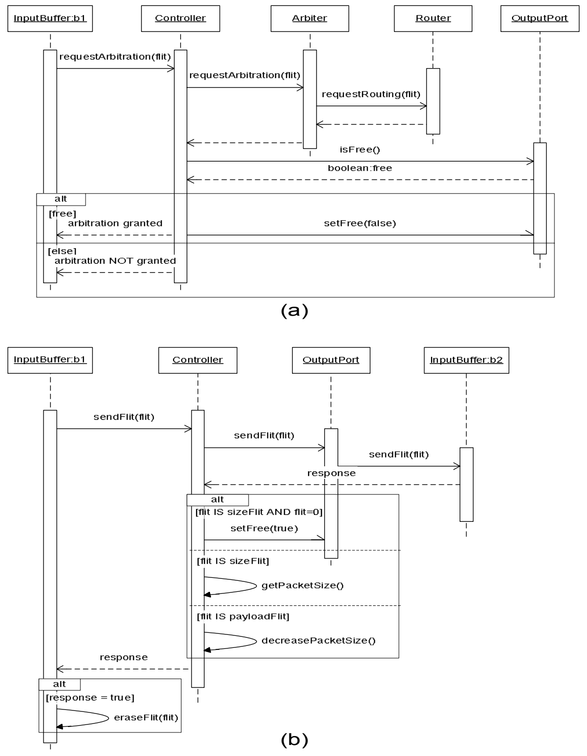


Fig. 2 – UML sequence diagrams depicting interactions between components of HERMES NoC.

4. NoC Modeling and Simulation

An actor-oriented model of HERMES was implemented using Ptolemy II, following the interaction descriptions shown in the previous section. This NoC model, named RENATO, was originally implemented without any timing information, so that it reflected only the functionality of the interconnect. To improve the accuracy of the simulation within Ptolemy II, timing information extracted from the simulation of the HERMES VHDL model was added to RENATO's corresponding transactions. Fig. 4 presents the RENATO model implemented on Ptolemy II.

The traffic load used for the performance comparison was generated by the Atlas framework [6]. Atlas was also used to generate the different HERMES configurations and to perform the analysis of the results. Two different NoC topologies were evaluated, 3x3 and 4x4 meshes, both with routers containing 8-position input buffers, handshake flow control, centralized arbitration and XY routing algorithm. The generated traffic has a transmission rate of 200Mbps, random destination nodes and the interval between packets followed a normal distribution. Fig. 3 shows the obtained results when comparing the average latency of all packets (normalized to HERMES clock cycles).

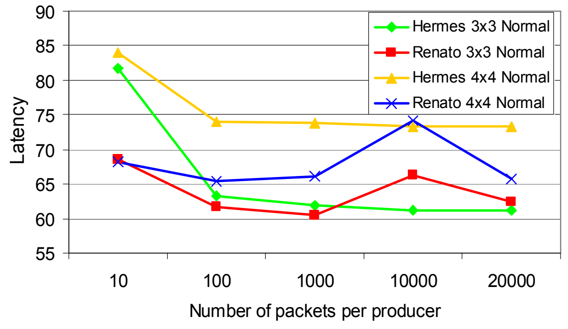


Fig. 3 – Comparison of average latency obtained by HERMES RTL simulation and RENATO actor-oriented simulation.

Table I shows the error of the average latency of RENATO in comparison to HERMES, in percentage. For long-lasting traffics, the error is in the order of 10%, which is a very good figure considering that the actor-oriented model is based only on interactions and works without the synchronization of a clock signal. That is critical in systems like a NoC interconnect, where small differences can cause a significant difference on the overall performance. For instance, if the arbitration request of one input buffer arrives to the arbiter slightly late, the latency of that packet can be increased in hundreds of cycles.

Table I – Average latency error between the models.

NoC size / Traffic	# of packets per producer	Average Latency HERMES	Average Latency RENATO	Average Latency Error (%)
3x3 - T1	10	81,72	68,46	16,22
3x3 - T2	100	63,32	61,62	2,68
3x3 - T3	1000	61,85	60,51	2,16
3x3 - T4	10000	61,1	66,22	-8,37
3x3 - T5	20000	61,09	62,44	-2,20
4x4 - T1	10	84,04	68,23	18,81
4x4 - T2	100	73,96	65,42	11,54
4x4 - T3	1000	73,85	66,07	10,53
4x4 - T4	10000	73,31	74,17	-1,17
4x4 - T5	20000	73,33	65,73	10,36

5. Improved observability and debugging

To improve the potential of observing and debugging the execution of an application running on top of a NoC architecture modeled according to the proposed approach, three different levels of debugging are provided:

(i) code-level debugging – supports debugging the internal functionality of individual actors implemented in Ptolemy II using regular code debugging tools like Eclipse. It is mainly used to verify if the sequential algorithm within a particular actor (i.e. routing) works properly;

(ii) interaction-level debugging – keeps track of the progress of the interactions. It uses textual output to inform the sending and receiving of each of the messages denoted in the sequence diagrams along with their timing information;

(iii) system-level debugging – extended observability resources were implemented as extensions to Ptolemy II. They can be plugged into different parts of the NoC model, working as probes which collect data from the network, process and display it graphically. They were combined on a small framework called NoCScope, which can analyze buffer occupation, power consumption on inter-router channels (by analyzing the transition activity on the channel, as previously done in [7]) and also the load on each channel. In all cases, the displays are updated as the simulation runs, so a designer can have full observability of the dynamic behavior of the NoC.

6. Conclusions

The major goal of this work is to support interactive design space exploration. This means that a designer should be able to analyze different alternatives for the on-chip interconnect, aiming to satisfy the particular requirements of performance, area, cost and power consumption for a given application domain. By following the presented approach, a designer can quickly change buffer schemes, routing and arbitration algorithms, and even the network topology, then simulate the model in Ptolemy II and obtain figures for performance, area and power consumption which are reasonably accurate, all within minutes. Obviously the figures can't be as detailed and accurate as the ones obtained by RTL simulation, but the loss of accuracy is acceptable for a system-level analysis as it can be compensated by the ease of change of the model and the simplicity on displaying the results.

To validate the accuracy of the proposed approach, the performance results of the actor-oriented model were compared with the results of a cycle-accurate simulation, showing errors in the order of 10% for long-lasting traffics, which are acceptable for design space exploration. Besides the performance analysis, the actor-oriented model can provide three levels of debugging and observability, including graphical information of buffer occupation, channel load and power consumption figures.

The proposed approach is not restricted to the HERMES NoC or to the Ptolemy II framework. It could be applied to each and every NoC (as long as its interactions can be captured as shown in Section 4.2) and the corresponding actor-oriented model could be built on any simulator supporting multiple models of computation.

7. References

[1] Moraes, F.; Calazans, N.; Mello, A.; Möller, L.; Ost, L. **HERMES: an Infrastructure for Low Area Overhead Packet-switching Networks on Chip**. Integration VLSI Journal, 38(1), 2004.

[2] Jie, L.; Eker, J.; Janneck, J.; Xiaojun L.; Lee, E. **Actor-Oriented Control System Design: A Responsible Framework Perspective**. IEEE Transactions on Control Systems Technology, 12(2), 2004.

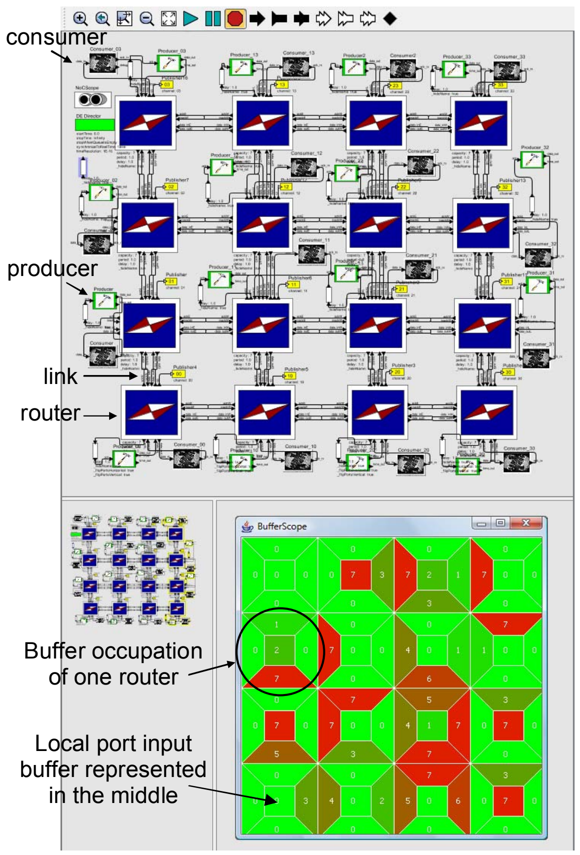
[3] Lee, E.; Neuendorffer, S.; Wirthlin, M. **Actor-Oriented Design of Embedded Hardware and Software Systems**. Journal of Circuits, Systems, and Computers, 12 (3), 2003.

[4] Knapp, A.; Wuttke, J. **Model Checking of UML 2.0 Interactions**. Lecture Notes in Computer Science 4364, 2006.

[5] Bjerregaard, T.; Mahadevan, S. **A Survey of Research and Practices of Network-on-Chip**. ACM Computing Surveys 38(1), 2006.

[6] Atlas NoC Framework. **http://www.inf.pucrs.br/~gaph/AtlasHtml/AtlasIndex_us.html**. 2007.

[7] Palma, J. C. S.; Indrusiak, L. S.; Moraes, F. G.; Garcia Ortiz, A.; Glesner, M.; Reis, R. **Adaptive Coding in Networks-on-Chip: Transition Activity Reduction Versus Power Overhead of the Codec Circuitry**. Lecture Notes in Computer Science 4148, 2006.



consumer
producer
link
router
BufferScope
Buffer occupation of one router
Local port input buffer represented in the middle

Fig. 4 – Ptolemy II showing a 4x4 NoC.