Model-based Design Flow for NoC-based MPSoCs

Luciano Ost[1], Leandro Soares Indrusiak[2], Sanna Määttä[3], Marcelo Mandelli[1], Jari Nurmi[3], Fernando Moraes[1]

[1] PUCRS, Av. Ipiranga 6681
Porto Alegre - Brazil
{luciano.ost, marcelo.mandelli, fernando.moraes}@pucrs.br

[2] Department of Computer Science
University of York - YO10 5DD
York - United Kingdom
lsi@cs.york.ac.uk

[3] DCS - Tampere University of Technology
P.O.Box 553, FIN-33101
Tampere - Finland
{sanna.maatta, jari.nurmi}@tut.fi

Abstract—**High-level abstraction modeling of NoC-based MPSoCs is an emerging approach to handle the vast design space alternatives of such systems. In this context, this work presents a model-based design flow, allowing the design space exploration of NoC-based MPSoCs at early stages of the design flow. The proposed flow supports accurate performance evaluation, latency and power consumptions for example, which are important to the adoption or rejection of design alternatives. A case of study is used to demonstrate some steps of the flow and to show some possible performance parameters that can be considered in the proposed design flow. (Abstract)**

Keywords: MPSoC; NoC; Actor Orientation; High Level Modeling; Design space exploration (key words)

I. INTRODUCTION

Due to the vast design space alternatives, evaluate the NoC-based MPSoCs at lower abstraction levels does not provide the required support to find out the most efficient NoC architecture considering the performance constraints (e.g. latency, power) of a given application at early design process stages. Thus, NoC-based MPSoCs design requires simple and accurate high-level models in order to achieve precise performance results, of each design alternative, in an acceptable design time [1].

In this perspective, the main *contribution* of this work is the integration of a set of tools and actor-oriented models into a model-based design flow developed into the Ptolemy II framework. The proposed flow enables flexible modeling and joint validation of application and platform models under different constraints, mappings, and configurations, allowing design space exploration of NoC-based MPSoCs at early design stages of the flow. Using this flow, designers can simplify the development and the validation of NoC-based MPSoCs, since inappropriate designs can be discarded (design space reduction) in a shorter time.

Figure 1 summarizes the present work, proposing a complete NoC-based MPSoC validation flow. *Application modeling* includes actors and executable UML sequence diagrams, to enable the description of actual embedded applications **(A)**. The application representation can be automatic converted to graph description (used for mapping propose) **(B)** **(C)**. Platforms, also described with actors, enabling fast design exploration and system debugging by using scope actors responsible to monitor some performance figures **(D)**. Performance parameters (e.g. latency, power and throughput) are generated during the joint validation of application model mapped onto the platform model **(E)**, keeping the accuracy of lower abstraction levels **(F)**.

This work is organized as follows. Section II presents related works in NoC-based MPSoC modeling approaches. Section III presents the proposed model-based design flow. Section IV includes a case study with the design space exploration of a NoC-based MPSoC running four applications. Finally, V points out conclusions and directions for future work.

II. RELATED WORK

Ha et al. [2] proposed a model-based framework for MPSoC software development, called HOPES. HOPES allows to model applications by using UML 2.0 and PeaCE model. The application model is based on actor-orientation and it can be specified with three different MoCs (Models of Computation). Once defined the application model, it is manually partitioned and mapped into the abstract PEs that compose the hardware platform (NoCs are not considered).

Pimentel et al. [3] present the Sesame (Simulation of Embedded System Architectures for Multilevel Exploration), which comprises three model layers: (*i*) *application model* using Kahn Process Network (KPN) to implement the application(s) behavioral; (*ii*) *mapping layer* that supports the application events traces mapping onto the PEs (applying dataflow graphs), and (*iii*) *architecture model* that defines architecture resources (NoCs are not considered) and captures their performance constraints (power is not considered) according to the computation and communication events generated by an application model.

Kangas et al. [4] present the Koski design flow that covers the design phases from system-level modeling to FPGA prototyping. Koski allows application and platform modeling using an UML 2.0 extension, targeting embedded real-time system design. The UML application model is defined according to a Kahn process network, while the architecture model is defined in UML composite structure diagrams according to the application model definitions. This approach uses a mapping model, based on the UML profiles, which defines the relationship between the application and the architecture models. Before the application mapping, tasks are grouped in blocks that are manually mapped (or pure random selection) onto the target architecture.

To the best of our knowledge, the present work is the first model-based design flow that covers the NoC-based design phases into the same flow by employing abstract and accurate models. The proposed approach employs a set of mapping heuristics, while the mapping process is merely manual in the reviewed works. In addition, the proposed approach allows accurate power analysis by using the rate-based power NoC model (in practice, an error of 0% when compared to the RTL simulation and 5% when compared to a commercial tool) [5]. This feature is another contribution of this work due to the fact that the rate-based power model considers low-level effects (congestion and burtiness) that are essential to verify the occurrence of hotspots. In the present work, the term hotspot refers to instants where power dissipation reaches a peak value, as analyzed in Section IV B.

III. MODEL-BASED DESIGN FLOW

Figure 1 presents the proposed model-based design flow. Below, each paragraph describes a given step of the flow.

Application Modeling: this is the level at which designers specify application blocks, implemented as a set of communicating actors (e.g. AB_1 and AB_2). Application blocks may contain one or more tasks that execute sequentially (for instance AB_4 with tasks *T1* and *T2*). Actors have input and output ports for sending and receiving messages. A UML Sequence Diagram specifies their exchange of messages. Actors are defined as active or passive and are represented as *lifelines* within one or more Sequence Diagrams [6]. *Active* actors are those that initiate a communication, while *passive* actors react or initiate a communication after receiving a message from an active one. By analyzing how often each actor executes and its patterns of communication, the model can characterize the processor workload imposed by the execution of each application block.

Application Graph and C code Generation: Ptolemy II was extended to parse the application model (e.g. UML Sequence Diagrams, message sizes) and generate a graph description according to a communication weighted graph model (CWG) used in CAFES mapping tool [7]. In this work, CWG nodes represent lifelines and edges represent the communication between them. Each edge contains a message (communication weight - total number of flits sent from a source to a target). In addition, it is possible to generate C codes of the modeled applications to be executed by the HEMPS MPSoC platform [8]. The generated C code can be executed according to the supported primitives of HEMPS, which adopts a communication API similar to MPI.

Mapping: the generated graph is used as input to the mapping tool, which maps application blocks onto PEs connected to a mesh NoC (e.g. AB_4 mapped onto PE connected to the bottom left router), as illustrated in Figure 1. Here, designers choose a mapping heuristics to define an application-platform mapping that can satisfy the application requirements [7]. Once the application-platform mapping is defined, the mapping tool generates two files. The first file is used as input by the *Mapper Actor*, which defines its *Mapping table* (structure that contains the PE position and its associated task). The second file is used by the HEMPS Editor Tool to define the mesh NoC dimension, as well as the mapping of the generated tasks (C code).

Platform and Scopes Selection: at this step, the platform is optimized according to the application requirements through parameterization and selection of scopes. NoC parameters include buffer depth, control flow, routing algorithm, and number of virtual channels. *Scopes* can monitor the power dissipation, hot-spots, throughput, latency, and buffer usage. For instance, the PowerScope is a parameterizable actor developed to display graphically the NoC power dissipation during simulation. It uses a monitor actor, called *Router monitor*, which is used to collect the average reception rate of each input buffer and the switching activity of its associated link. By means of the PowerScope, the total NoC power dissipation (and energy consumption) is calculated and displayed during the simulation. PowerScope generates graphics (e.g. Figure 3), and a report of energy consumption, maximum, minimum and average power per router.

PtolemyII Simulation: once defined the application model and its mapping onto the selected platform model, the design is validated through abstract simulation, which allows the generation of performance figures.

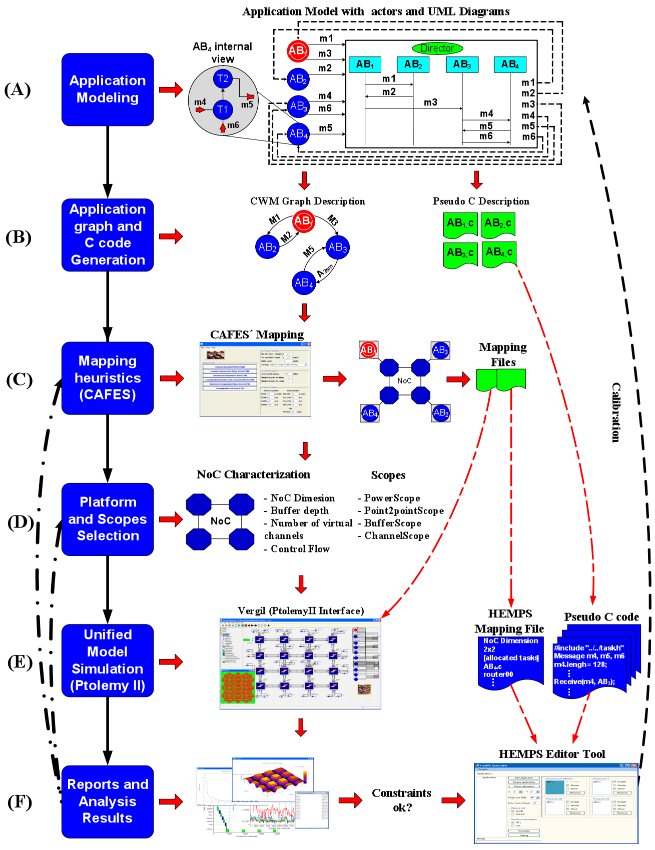

Figure 1 - Proposed model-based design flow.

Reports and Analysis Results - designers evaluate the impact of a given platform on the application (unified solution) by evaluating graphs and reports generated by the selected scopes, to verify if the chosen NoC architecture fulfills the power and performance requirements. Furthermore, *iff* the resulting solution does not satisfies the performance requirements (e.g. end-to-end communication latencies are not met) imposed by the target NoC-based MPSoC, the designer may re-define some structural parameter (e.g. routing algorithm, buffer depth) or even re-mapping the lifelines by applying another mapping heuristic. Once one possible good configuration regarding application-mapping-platform is taken, the generated C code can then be executed in the HEMPS platform (RTL cycle accurate). This allows designers to extract important performance figures, which are not considered into the unified model. For instance, the real load imposed by each task execution. These values can be used to re-calibrate each layer of the unified model in order to evaluate different solutions that can improve the performance of the target NoC-based MPSoC.

IV. Experiments and Results

This Section illustrates the use do the proposed design flow. Two performance metrics of a NoC-based MPSoC are evaluated: (*i*) end-to-end communication latency; (*ii*) power estimation (average power dissipation and hotspots). Such metrics are obtained for different application characteristics and two different mapping heuristics. The simulation scenario has all four applications executing simultaneously in the same platform for one second and the design space exploration varies:

- switching activity in the NoC links: 10%, 20%, 30%, 40% and 50%;
- two mapping heuristics: random (reference worst-case mapping), GI (greedy incremental).

Four applications were modeled (using actors and UML diagrams) in Ptolemy II: (*i*) *HDTV*, comprising end-to-end transmission of 10 HDTV channels, modeled as 2 application blocks, with frames obtained from real application traces; (*ii*) VOPD (*Video Object Plan Decoder*), modeled as 12 application blocks; (*iii*) *MPEG4 decoder*, modeled as 12 application blocks [1]; and (*iv*) *an automotive application,* modeled as 10 application blocks [6].

The NoC infrastructure has the following parameters: 6x6 mesh topology, XY routing algorithm, 32-bit flit size, packets with 128 flits and handshake control flow. The rate-based power model was calibrated using the XFAB XCMOS 0.18 µm (XC018) 1.8V technology, adopting clock-gating, and a 250 MHz clock frequency.

A. End-to-end Communication Latency Analysis

Table I presents the maximum (Max.), minimum (Min.) and average (Av.) end-to-end communication latency values for the VOPD application. The end-to-end communication latency is defined here, as is the delay between the time a PE starts its message transmission and the time the target PE receives the message. Results show that the use of mapping heuristic (GI, in this example) reduces the end-to-end communication latencies. For example, taken the end-to-end communication presented in the second row (Vopme => VOPrec), it is possible to verify a reduction of 32,7%, 4,1% and 6,11%, for maximum, minimum and average latency, respectively, when comparing the GI to random mapping.

TABLE I. VOPD end-to-end communication latency results for two different mapping heuristics. Application (A), maximum (Max.), minimum (Min.) and average (Av.). Values in clock cycles.

Heuristics		Greedy Incremental			Random			Difference		
A.	Comm. Name	Max.	Min.	Av.	Max.	Min.	Av.	Max.	Min.	Av.
VOPD	Iquant > IDCT	1188	834	851,4	980	834	844,65	208	0	6,77
	Vopme > VOPrec	1093	1085	1085,2	1524	1120	1162,26	-431	-35	-77,00
	StripeM > IQuant	178	98	100,58	207	119	132,74	-29	-21	-32,16
	VOPme > Pad	248	248	248,00	443	255	298,23	-195	-7	-50,23
	ACDC > IQuant	1299	849	965,71	1333	849	979,06	-34	0	-13,35
	Iscan > ACDC	807	807	807,00	1002	850	869,35	-195	-43	-62,35
	UPsamp > VOPrec	741	725	726,68	1552	718	1107,16	-811	7	-380,48
	Pad > VOPme	730	724	724,58	953	731	741,58	-223	-7	-17,00
	IDCT > UPsamp	835	821	821,90	834	814	816,26	1	7	5,65
	VOPrec > Pad	741	731	731,39	791	731	734,48	-50	0	-3,10
	ARM > IDCT	97	91	91,39	114	112	112,13	-17	-21	-20,74
	ACDC > StripeM	175	171	171,19	160	150	151,94	15	21	19,26
	Run > IScan	807	807	807,00	1354	842	894,06	-547	-35	-87,06
	VLD > Run	165	159	161,19	177	175	175,06	-12	-16	-13,87

B. Power Analysis

Figure 2 shows the NoC average power dissipation for two different mapping heuristics (random and GI), when varying the link switching activity. The impact of the mapping heuristic on the average power dissipation can be clearly observed. The power dissipation increases with the increase of the switching activity: for a switching activity of 50%, the difference between GI and random mapping reaches 48.16%. Such results show the importance of profiling an application's average switching activity, and using it to guide design choices. The heuristic GI presents smaller NoC average power dissipation because applications are mapped in one region, minimizing network congestion.

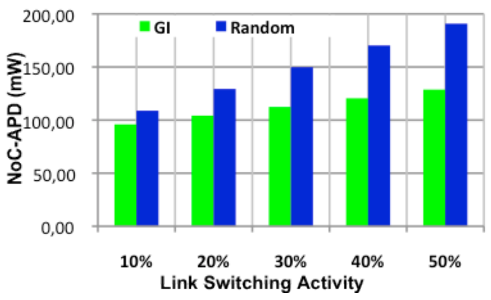

Figure 2 - NoC average power dissipation for two mappings and link switching activity.

Another feature of the proposed approach is the possibility to verify the occurrence of hotspots, which can impact the performance of the whole system and even reduce its reliability and lifetime. Due the flexibility of the proposed approach, the user can define an interval of instantaneous power dissipation (IPD) values (or a set of it) according to the design requirements. A hotspot is defined, is this work, as an IPD value superior to 3 times the NoC average power dissipation. Figure 3 presents the hotspot occurrences, in the defined interval, during 1 second of simulation.

The random mapping heuristic has 5806 hot-spot occurrences in the defined interval (one second simulation), with a 1544,37 mW peak. By employing the GI mapping heuristic, only 289 occurrences are observed in the same interval (reduction of 95.02%), with a 608,90 mW peak. Note that the impact of the mapping heuristic is more expressive in terms of number hotspots than in terms of NoC average power dissipation.

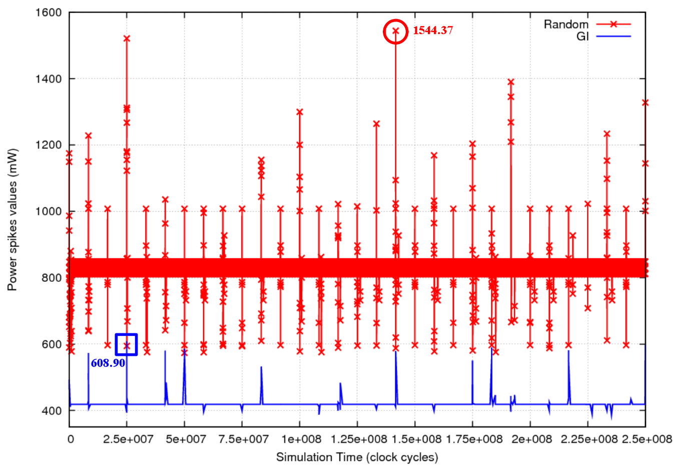

Figure 3 - Power values in defined interval with 50% of link switching activity, during 1 second of simulation.

The flow also generates the power spatial distribution, as illustrated in Figure 4. This Figure shows the hotspot communication zones at a specific simulation time (e.g. when the NoC achieves its peak power value).

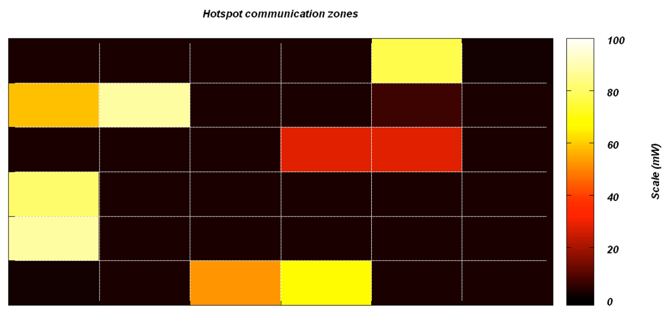

Figure 4 - Local analysis of hotspot communication zones at the peak power value (GI – 608.90 mW).

Finally, it is important to mention that in terms of simulation time the proposed approach is about 5 to 6 times faster than the HEMPS platform (using ISS - instruction set simulators and C/SystemC models). For instance, by executing two applications (VOPD and MPEG4) in a 5x5 HEMPS platform during one second, the total execution time is 17 hours (not considering the power estimation). On the other hand, for simulating the same scenario by using Ptolemy II simulation requires less than 3 hours (considering the power estimation).

V. Conclusion

This work presented a model-based methodology and a supporting toolset that enable the design space exploration of NoC-based MPSoCs at early stages of the design flow, when design decisions are actually taken. It uses an actor-oriented simulation framework that captures the dynamic behavior of the NoC components and feeds its parameters to a group of Scopes, providing accurate performance evaluation. The accurate performance evaluation is achieved by calibrating the proposed high level models using a reference design model, for instance an RTL implementation, as shown in [5].

Due to the flexibility of the proposed approach, complete separation between application and platform models, designers do not have to restrict themselves to a single platform template (as done in this work), and they can successively evaluate the performance of an application running over different platforms (already supported or that can even be implemented and integrated into the proposed model-based design flow), allowing extensively exploration. This flexibility can significantly reduce the design time, starting from system-level specification and going down to a more detailed implementation (e.g. HEMPS that was used in this work).

By integrating the rate-based power estimation method into the proposed design flow, it is possible to obtain accurate NoC power results (e.g. hot-spots, peak power values) of multi-applications mapped onto NoC-based MPSoCs platforms. These results could not be obtained using current volume-based models, since they do not consider NoC low-level effects. Presented experiments show how the proposed model-based flow can support designers on the evaluation of mapping heuristics, aiming to reduce end-to-end communication latency, power and the occurrence of hot-spots.

Acknowledgements

This research was supported partially by CNPq (Brazilian Research Agency), project 301599/2009-2.

References

[1] Lee S. E. at. al. **A high level power model for Network-on-Chip (NoC) router**. Computers & Electrical Engineering, 35(**6**), 2009.

[2] Ha, S. **Model-based Programming Environment of Embedded Software for MPSoC**. In: ASP-DAC'08, 2008.

[3] Pimentel, A. D. ; et. al. **Calibration of abstract performance models for system-level design space exploration**. Journal of Signal Processing Systems, v. 50(**2**), 2008.

[4] Kangas, T.; et. al. **UML-based multiprocessor SoC design framework**. ACM Transactions on Embedded Computing Systems, v. 5(**2**), 2006.

[5] Ost, L. et al. **A high abstraction, high accuracy power estimation model for networks-on-chip**. In: SBCCI'09, 2009.

[6] Määttä, S. et al. **Validation of Executable Application Models Mapped onto Network-on-Chip Platforms**. In: SIES'08, 2008.

[7] Marcon, C.; et. al. **Comparison of network-on-chip mapping algorithms targeting low energy consumption**. IET Computers and Digital Techniques, 1(**2**), 2008.

[8] Carara, E.; Oliveira, R.; Calazans, N.; Moraes, F. **HeMPS - A Framework for NoC-Based MPSoC Generation**. In: ISCAS'09, 2009.