Exploring Dynamic Mapping Impact on NoC-based MPSoCs Performance Using a Model-based Framework

Luciano Ost[1], Marcelo Mandelli[2], Gabriel Marchesan Almeida[1], Leandro Soares Indrusiak[3], Leandro Moller[4], Manfred Glesner[4], Gilles Sassatelli[1], Michel Robert[1], Fernando Moraes[2]

[1] LIRMM – 161 rue Ada, Cedex 05 - 34095 Montpellier, France
{ost, marchesan, sassatelli, michel.robert}@lirmm.fr

[2] FACIN-PUCRS - Av. Ipiranga 6681- 90619-900, Porto Alegre, Brazil
{marcelo.mandelli, fernando.moraes}@pucrs.br

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

[4] FG MES – Technische Universität Darmstadt - Karlstr. 15, 64283 Darmstadt, Germany
{moller, glesner}@mes.tu-darmstadt.de

ABSTRACT

The power evaluation of NoC-based MPSoCs is an important and a time-consuming process. Mapping tasks onto processing elements (PEs) has a critical impact on system performance, as well as power dissipation. To cope with complex dynamic behavior of applications, it is common to perform task mapping at runtime so that the utilization of processors and interconnect can be taken into account when deciding the most appropriate PE to host tasks. On the other hand, the process of accurately comparing different mapping heuristics can be very costly once each adopted solution has to be evaluated using simulation that can take hours or even days in the case of large MPSoCs. In this context, this paper has two major contributions: (i) evaluation of dynamic mapping by employing a model-based framework that unifies abstract models of applications, NoC-based platforms and mapping heuristics, and (ii) power consumption evaluation of such heuristics by using a rate-based power model.

Categories and Subject Descriptors

B.7.1 [**Integrated Circuits**]: Types and Design Styles – advanced technologies, VLSI (very large scale integration).

General Terms

Design, Experimentation, Performance, Verification.

Keywords

Mapping Heuristics, Power Modeling, NoC-based MPSoCs

1. INTRODUCTION

Portable embedded systems have limited power budget that must be efficiently used [1]. Due to the large simulation time and amount of memory required by the power estimation tools, simple and accurate high level models became necessary to achieve acceptable power results within the time-to-market frame of future portable NoC-based MPSoCs. These systems are composed of multiple PEs, dedicated hardware and software components that are interconnected by a Network-on-Chip (NoC) [2]. HDTV, multiple wireless communication standards, and media players are examples of applications, which can be executed in such systems, requiring high performance allied to low power consumption. These applications can have unpredictable behavior, due to the variability of the inter-task communication patterns (e.g. different data rates) [3]. Besides, the workload of such systems may vary during the execution time due to user (e.g. initialization of new applications at runtime) and/or environmental parameters (e.g. change the frequency operation for optimizing battery lifetime) requirements [4][5][6].

A common runtime technique to cope with the unpredictable behavior of applications is to define the place of each task at the execution time. The tasks placement is defined based on dynamic mapping heuristics, which employ different criteria (e.g. distance between tasks). However, for analyzing the mapping behavior and its impact on NoC-based MPSoCs performance becomes a critical issue, since designers should be able to simulate scenarios for long time where the occurrence and analysis of critical aspects (e.g. application area fragmentation) would be possible to be observed. In this context, it is necessary to provide flexible approaches whereby a designer can set up large scenarios that can be easily extrapolated in order to predict the impact imposed by different dynamic mapping heuristics. Furthermore, these approaches must be able to produce performance metrics (e.g. latency, power consumption) in a short amount of time, allowing the elimination of not suitable design alternatives.

In this scenario, this paper covers the integration of a dynamic mapping heuristics into an unified model of a NoC-based MPSoC, enabling fast design space exploration, while proving accurate performance metrics imposed by each design alternative (e.g. mapping heuristic). Another contribution of this work is the use of a rate-based power model that allows the evaluation of the dynamic mapping in terms of hotspots. This term refers to instants where power dissipation reaches a peak value, which may increase the temperature at specific regions of the chip that can, consequently, generate hotspots.

This paper is organized as follows. Section 2 describes related

SBCCI'11, August 30–September 2, 2011, João Pessoa, Brazil.

works in high-level model-based frameworks and dynamic task mapping. Section 3 summarizes the basic aspects of the adopted unified model-based framework, as well as a set of dynamic mapping heuristics and their integration into the unified model-based framework. Results, including model accuracy and hotspots evaluation are presented in Section 4. Finally, Section 5 points out conclusions and directions for future work.

2. RELATED WORK

Ha [7] proposes a model-based framework for MPSoC software development, called HOPES that allows the modelling of applications by using UML 2.0 and PeaCE model. Once the application model (based on actor-orientation) is defined, it is manually partitioned and mapped into the abstract PEs that compose the hardware platform. This model does not consider the use of NoCs as interconnect.

Pimentel et al. [8] present the Sesame, which comprises three model layers: (*i*) *application model* using Kahn Process Network (KPN) to implement the application(s) behavior; (*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.

Two other approaches use KPN to model the application behavioural [9] and [10]. In [9], KPN application model and abstract platform templates are used to automatically generate executable and timing TLM descriptions of MPSoCs platforms. The mapping process is defined at design time and it is limited to selecting the best fit of KPN-based applications to platform modules according to the pre-existing UML template information. In turn, Kangas et al. [10], uses a UML profiles to map KPN-based applications models onto the platform model. Before the application mapping, tasks are grouped in blocks that are manually mapped (or randomly selected) onto the target platform. Both application and platform models are defined according to an UML 2.0 extension, targeting embedded real-time system design.

The approaches above are flexible enough for modeling and validating several applications mapped onto MPSoC platforms models. Furthermore, such approaches can achieve a good simulation performance. However, the mapping process exploration is limited to manual or static decisions, which are insufficient to handle the dynamic and unpredictable behavior of future embedded applications. Thus, several dynamic mapping heuristics have been proposed [3][4][5][10][11][13][14][17]. Such heuristics aiming to satisfy QoS requirements (e.g. [10][12]), to optimize resources usage (e.g. network contention [3][13][14]), and to minimize the energy consumption (e.g. [3][5][11][12] [13][17]). Each of them has its own parameters and cost functions, which can have numerous of variations (defined as new heuristics with different properties and strengths). Thus, choosing the optimal dynamic heuristic for a set of applications is not trivial; hence to evaluate different performance metrics is time-consuming. A common optimization metric is the energy consumption evaluation based on the volume-based power model, which does not consider low-level effects (e.g. network contention), which are essential to verify the occurrence of hotspots [21]. Hotspots can impact the performance of the whole system and even reduce its reliability and lifetime [15][16].

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 application-mapping-platform models. The fundamental principle behind the unified model-based framework is the complete separation between the different layers – application, mapping and platform. Therefore, new application models, platform templates and mapping heuristics can be integrated to the framework as long as they follow the pre-defined inter-layer APIs [21]. In this context, a set of dynamic mapping heuristics were incorporated to our approach, allowing exploring the fundamental behavior inherent to dynamic mapping heuristics, while providing design flexibility and high debugging capacity. In addition, excluding the work presented by Singh [3], only mono-task dynamic mapping heuristics are proposed and evaluated in the literature. In this context, this work extends two dynamic mapping heuristics to support multi-task assigned per PE. Finally, the present work is the first to employ the rate-based instead of using the volume-based power model (common decision of reviewed works) to evaluate the impact of different dynamic mapping heuristics on NoC power dissipation and on occurrence of hotspots.

3. UNIFIED MODEL-BASED FRAMEWORK

The design space exploration of MPSoCs can be divided into three main layers: (*i*) application, (*ii*) mapping[1], and (*iii*) platform. The first layer comprises application modeling and validation (functionality and requirements), while the second layer defines how such applications are mapped onto the MPSoC platform (third layer).

3.1 Application Layer

Most researchers on dynamic mapping use task graphs to model applications. This approach does not support the necessary flexibility to develop and to validate complex and distributed applications used in present MPSoCs. In this context, the current approach provides the software engineer with the possibility of developing and validating different applications regarding only their functionality and requirements by using executable models based on UML sequence diagrams and actor-orientation, as proposed in [18].

UML is a well-known standard modeling language used by most part of the software development industry due to its flexibility, support to the real time requirements through profiles and tool support. On the other hand, actor oriented design is a component methodology, which separates the functionality concerns (modeled as actors) from the component interaction concerns (modeled as frameworks). It includes the definition of the execution semantics as a part of the model rather than of the underlying simulation engine [19]. As a result, the concurrent behavior and the interdependencies of the application tasks can be captured more accurately. We claim here that the combination of UML and actor-orientation provides more design flexibility to specify the application tasks, their dependencies, synchronization mechanisms, and data exchanges, when compared to application modeling approaches based on task graphs. For instance, to model the communication behavior of a given application by employing sequence diagrams is more intuitive than using task graphs, since

[1] In the context of this work, the mapping layer is a behavior entity called *Mapper* that is characterized according to a set of operations, which is used to define task mapping during the simulation.

combined fragments and interaction operators, such as option, parallel, loop can be used.

3.2 Platform Layer

As the number of PEs grows, the design of NoC-based MPSoCs becomes increasingly communication centric. In this context, one of the main challenges in the future MPSoC projects will be the communication infrastructure, representing up to 30% of the total power consumption of the whole system. The present approach provides a set of models that differ according to its accuracy and its required simulation time. It provides multi-accuracy platforms models (e.g. latency, throughput and power estimation), allowing designers to choose between faster or more accurate validation, as they require.

The present framework provides adequate possibilities for observing and debugging the execution of a set of applications running on top of NoC-based platforms. In this context, the platform model layer includes Scope actors that can be used to check the running status of the system, as well as to collect performance figures that can be used for application/platform model optimization. Examples of Scopes are the LatencyScope and the PowerScope. The LatencyScope, provides end-to-end[2] communication latency figures for each task communication.

The PowerScope generates power reports based on volume-based [20] and/or rate-based [21] NoC power models. The volume-based model estimates the average power as a function of the total transmitted data. This model evaluates the energy consumption in an end-to-end transmission only. Equation 1 describes the model energy consumption estimation, for a single data transmission between two points of the NoC.

$$E_{bit}^{hops} = n_{hops} \times E_{S_{bit}} + (n_{hops} - 1) \times E_{L_{bit}} \qquad (1)$$

In the equation above, E_{SBit} is the energy consumption in one router; E_{LBit} is the energy consumption of the interconnection wires; and n_{hops} is the number of routers used in the data bit transmission.

Therefore, the volume-based model does not capture low-level effects, such as congestion and burstiness; hence, they are simple but not accurate. On the other hand, models derived from electrical simulation are accurate but too complex to be integrated into abstract models. In turn, the rate-based power model constitutes a trade-off between such approaches: data volume is considered but computed as a transmission rate inside a given sample period, and accuracy is guaranteed from a physical calibration step, which defines the power dissipation for each router component and transmission rate. Such models are highly customizable and can easily be applied to different NoC architectures and technologies. Furthermore, the results produced by the PowerScope, when employing the rate-based power model, have in practice, an error of 0% when compared to the RTL simulation and 5% when compared to a commercial tool [21].

3.3 Mapping Layer

The mapping layer is the link between the application and the platform layer. It provides the necessary support to explore the mapping influence in terms of system performance, by employing a Mapper Actor that has a set of supported mapping heuristics, as well as the MapperScope. The MapperScope, is used to monitor the mapping layer (e.g. capturing each task requesting time) and to generate mapping figure, like number of hops among communicating tasks.

The proposed approach supports static (e.g. taboo search) and dynamic mapping heuristics, which were incorporated in order to support the evaluation of dynamic mapping behavior and its impact on NoC-based MPSoCs performance.

3.3.1 Integrated Dynamic Mapping Heuristics

Three dynamic mapping heuristics were integrated into the Mapper Actor: (*i*) nearest neighbor (NN), dependencies-neighborhood (DN), and (*iii*) lower energy consumption based on dependencies-neighborhood (LEC-DN).

Due to its simplicity, the nearest neighbor (NN) is used as reference mapping heuristics for dynamic single-task. The NN mapping considers only the proximity of an available resource to execute the required task. NN starts searching for a free PE able to execute the task near to the requesting task. The search tests all n-hop neighbors, *n* varying from 1 to the NoC size in a spiral way, stopping when the first PE free is found.

Differing from the NN heuristic, which tries to map the requested task as closely as possible to the requesting task, the DN heuristic considers all dependencies between tasks, mapping the requested task as closely as possible to the already mapped in which it communicates with, by employing a proximity (in number of hops) cost function.

The LEC-DN heuristic extends the DN by employing two cost functions: (*i*) proximity, in number of hops; (*ii*) communication volume among tasks (which corresponds to the communication energy). The second criterion is used when a given task communicates with at least two mapped tasks. In this situation, the new task is mapped closer to the task with higher communication volume. When the requested task has only one communicating task already mapped, LEC-DN uses the NN search method (spiral search). If there is more than one communicating task already mapped, the LEC-DN searches for a PE inside the bounding box defined by the position of such tasks [17].

Consider the application illustrated in Figure 1(a), containing 4 tasks, where AB_1 and AB_2 are initial tasks.

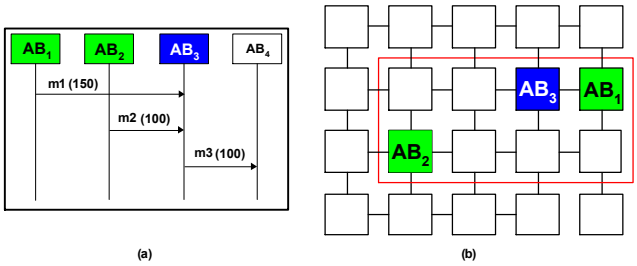

Figure 1 - (a) application described as UML diagram; (b) search space to map the AB_3 task, and one possible mapping for AB_3.

The mapping of task AB_3 is triggered by the first communication with it. The search space to map task AB_3 corresponds to the bounding box defined by the position of AB_1 and AB_2 tasks (Figure 2(b)). Thus, AB_3 will be mapped nearest to task AB_1, since according to the application graph the communication volume

[2] This term 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.

$AB_1 \rightarrow AB_3$ is higher than $AB_2 \rightarrow AB_3$. Note that task AB_4 is not mapped, since it depends from task AB_3.

The LEC-DN was implemented and validated in a RTL homogeneous NoC-based platform [22]. This implementation is adopted as reference model in this work (as explored in Section 4). In this implementation, the bounding box search method used in LEC-DN employs the volume-based energy model to select the requested task position (task to be mapped). This volume-based search method approach was kept in our implementation, since our first main goal was to integrate and to verify possible loss of accuracy, when comparing a real implementation [22] to the proposed approach.

The diagram illustrated in Figure 2, shows the mapping process execution implemented into the Mapper Actor. When the Mapper Actor receives a mapping request of a task t_i, it verifies if it task is already mapped and then starts executing the mapping flow. First, it checks if there is some available PE in the system. If there is no available PE, the requested task is scheduled to be mapped later. On the other hand, the flow proceeds to the next step. The next step verifies if the required task (t_i) is already pre-mapped. In an affirmative case, the task is allocated to the assigned PE; otherwise, the LEC-DN mapping heuristic is executed.

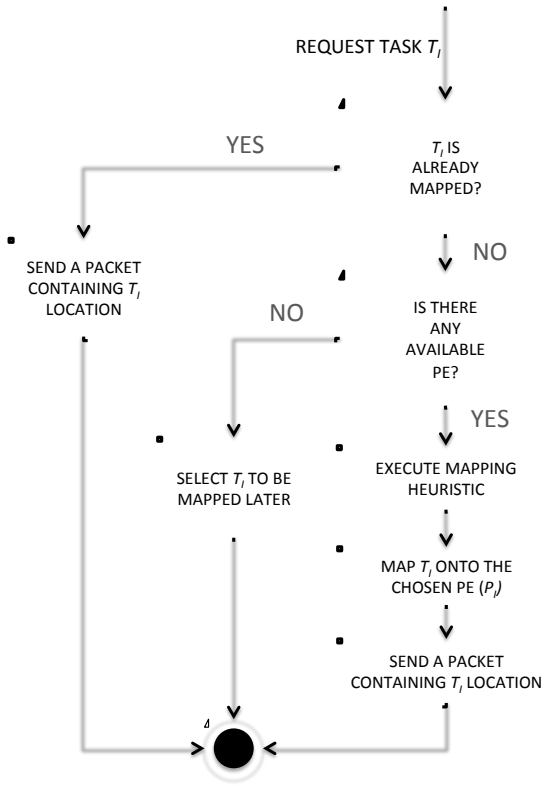

Figure 2 - Integration of the LEC-DN into the *Mapper Actor*.

The DN and LEC-DN heuristics were extended in this work to multi-task mapping. Basically, the heuristics start verifying if the requesting task PE (0 hops distance) is able to map the required task instead of looking for the neighbour PEs at 1 hop distance. Besides, both heuristics adopt the search method (bounding box). The LEC-DN differs from the DN because it uses the energy consumption as cost function.

4. RESULTS

4.1 Comparison between HeMPS and unified-model

This Section compares three mono-task dynamic heuristics: (*i*) NN, (*ii*) DN, and (*iii*) LEC-DN, using the HeMPS and unified-model. Both implementations were compared according to two metrics: (*i*) energy consumption (based on Hu's approach [18]), and (*ii*) number of hops. Such metrics were obtained for different application scenarios:

- Case A: VOPD (Video Object Plan Decoder) and MPEG4 decoder, both containing 12 application blocks and transmitting 30fps each, and an Image Segmentation, modelled as 6 application blocks;
- Case B: VOPD, MPEG and a MWD displaying 30fps;
- Case C: MPEG, Image Segmentation, MWD and a synthetic application composed of 4 application blocks;

Each application has at least one initial task that is manually mapped to any free PE, excluding the PE in position 33, which was reserved in this work to the Mapper Actor. In both implementations all applications execute simultaneously for one second. The platform setup used to evaluate the dynamic mapping heuristics is configured as follows: 2D-mesh topology, XY routing algorithm, 32-bit flit size, packets with 128 flits and a 7x6 (41 Slave-PEs one position reserved to the mapper) MPSoC dimension. The energy model was calibrated using the ST/IBM CMOS 65 nm technology at 1.0 V, adopting clock-gating, and a 100 MHz clock frequency.

Table 1 presents the difference in the average energy consumption and in number of hops between the unified-model (Section 3) and the RTL HeMPS platform. In average, the energy consumption difference between both models is 10%. In terms of number of hops, the results showed an average difference of 12.17%. The worst-case difference is presented in case B, where the difference on the average energy consumption to deliver all packets is 19,21%, while the difference in number of hops is 20.22%.

Aspects that are not considered in the unified-model justify the difference between both models. Such aspects influence the time that mapping requests arrive at the mapper, which may change the task mapping order (changing the number of hops and, consequently the energy consumption). Examples of these aspects are the OS execution time for each task (e.g. scheduling algorithm) and the heuristic execution time (altering the initial execution time of each task). The absence of these aspects do not invalidate the use of our approach to evaluate dynamic heuristics mapping, since all heuristics are equally analysed. Besides, the unified-model can be calibrated with values extracted from (e.g. task workload,) a target platform (e.g. HeMPS), allowing to achieve more accurate results. The unified-model, w.r.t the HeMPS model is approximately 34 times faster than the HeMPS platform (using ISS - instruction set simulators and C/SystemC models). For instance, by executing each case (A, B, and C) in the HeMPS platform with one-second duration, the total execution time is 17 hours, while the same scenarios by using our unified-model requires in average 30 minutes for each simulation. Considering that real embedded system applications may run up to dozen of seconds, we claim that the adoption of the present approach increased evaluation speed, since different heuristics and application's scenarios can be simulated in less time.

Table 1 - Energy consumption (EC) and number of hops (#H) difference (Diff.) between HeMPS and unified-model, using NN, DN and LEC-DN dynamic mapping heuristics considering cases A, B and C.

		NN			DN			LEC-DN		
		HeMPS	unified-model	Diff. (%)	HeMPS	unified-model	Diff. (%)	HeMPS	unified-model	Diff. (%)
EC (nJ)	case A	1.638E+04	1.726E+04	5.09%	1.657E+04	1.928E+04	14.08%	1.527E+04	1.699E+04	10.13%
	case B	1.344E+04	1.664E+04	19.21%	1.437E+04	1.590E+04	9.66%	1.384E+04	1.590E+04	12.95%
	case C	1.122E+04	1.166E+04	3.77%	1.172E+04	1.241E+04	5.58%	1.102E+04	1.241E+04	11.20%
#H	case A	79	81	2.47%	91	95	4.21%	77	84	8.33%
	case B	71	89	20.22%	76	93	18.28%	76	93	18.28%
	case C	81	88	7.95%	83	94	11.70%	77	94	18.09%

4.2 Hotspots Evaluation

Another feature of the proposed approach is the possibility of using the rate-based power model to verify the occurrence of hotspots, which may be produced by not optimized mapping decisions. The rate-based power model was calibrated using the XFAB XCMOS 0.18 µm (XC018) 1.8V technology, adopting clock-gating, and a 100 MHz clock frequency. The user can define an interval of instantaneous power dissipation (IPD) values (or a set of it) according to the design requirements.

Table 2 presents the relative power distribution (RPD) according to the NoC APD for a switching activity of 50%. The power distribution includes five intervals:

- interval 1, instantaneous power dissipation (IPD) between 25% and 50% times the NoC APD;
- interval 2, IPD between 50% and 75% times the NoC APD;
- interval 3, IPD between 75% and 100% times the NoC APD;
- interval 4, IPD between 100% and 200% times the NoC APD, and
- interval 5, IPD above 200% times the NoC APD.

In this work, intervals 3, 4 and 5 are considered here as hotspots. Besides, cases A, B and C, the following scenarios were evaluated:

- Case D: MPEG, Automotive, synthetic, multimedia applications containing 12, 10, 9 and 8 applications blocks, respectively.
- Case E: MPEG, MWD, Automotive, multimedia, synthetic and VOPD.

Due the flexibility of the present approach, scenarios where different number of application can be loaded during the execution time (new applications are requested when a number of pre-defined applications are already executing) could be evaluated. Thus, in case E, applications are executed during 3 seconds, where MPEG and MWD started at moment 0, one second later the automotive and the multimedia application are loaded in the system (emulation of a user request), and finally, one second further the synthetic and the VOPD applications are loaded. This scenario employs a 5x5 NoC-based MPSoC dimension.

Results show that all heuristics produce power dissipation peak value within RPD intervals 1 and 2. It's also worth noting that NN and DN heuristics present similar average power dissipation compared to LEC-DN, but the mappings produced by them result in more hotspot. For instance, in case E the power dissipation values for DN and LEC-DN are 51.05 and 51.31 mW, respectively. However, in the same case, the number of hotspots is 47,5% smaller when DN is employed. Regarding the mono-tasks cases (A, B, C, D, and E), the LEC-DN (2018) obtains the best hotspots results, when comparing to the NN (2485, reduction of 18,7%) and DN (2075, reduction of 2.8%). Due the similarity of results between DN and LEC-DN, only both heuristics were extended to multi-task and compared using the case E.

Figure 3 shows the number of hotspot occurrences in a 1s simulation duration. The LEC-DN heuristic produces 2324 hotspots peaks between intervals 4 and 5 (for the sake of simplicity not all are displayed), with a 168.57 mW peak. In turn, the DN heuristic produces 2671 hotspots peaks in the same interval, with a 138.07 mW peak. LEC-DN produces a smaller number of peaks in the intervals 4 and 5, but in interval 3 (also considered as hotspot zone) the number of hotspots peaks is almost 3 times higher than the ones produced when the DN heuristic is employed.

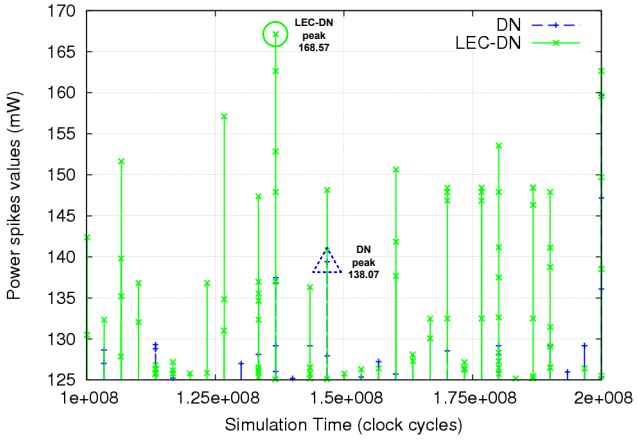

Figure 3 –Power values in hotspots intervals, after loading two applications onto a platform that is already running two other applications, during 1 second of simulation.

It is possible to observe that DN heuristic performs better for multi-task approach. Considering five different scenarios, it has been identified 12874 hotspots peaks against 14613 for LEC-DN heuristic. These results can be explained by the fact that when using DN heuristics, the mapper tries to map communicating tasks into the same PE as long as possible. On the other hand, LEC-DN heuristic considers all tasks' dependences and may spread tasks onto the MPSoC.

Table 2 - Number of hotspot peaks (# PICK) for different intervals using NN, DN and LEC-DN dynamic mapping heuristics considering cases A, B, C, D and E.

		NN					DN					LEC-DN				
		25-50%	50-75%	75-100%	100-200%	> 200%	25-50%	50-75%	75-100%	100-200%	> 200%	25-50%	50-75%	75-100%	100-200%	> 200%
# PICK	case A	785	944	405	19	4	743	813	280	328	1	880	1000	205	82	0
	multi	-	-	-	-	-	29	83	90	603	83	57	106	76	570	71
	case B	599	833	378	414	1	699	858	324	334	0	699	858	324	334	0
	multi	-	-	-	-	-	29	98	89	473	211	36	87	105	512	162
	case C	1133	255	33	13	0	1151	270	25	14	0	1153	270	24	10	0
	multi	-	-	-	-	-	169	119	227	240	22	149	228	115	224	28
	case D	4352	1481	418	74	0	6119	1248	376	54	0	6127	1436	385	48	0
	multi	-	-	-	-	-	4491	2919	4808	1979	151	3959	3064	4667	2378	294
	case E	6658	2172	579	145	0	3733	1705	277	72	0	6716	1447	506	110	0
	multi	-	-	-	-	-	12563	3899	1230	2432	239	14147	3360	3087	2119	205

5. CONCLUSIONS AND FUTURE WORKS

RTL NoC-based MPSoC modeling provides accurate results, while limits the evaluation of mapping heuristics due to the large simulation time. Thus, flexible, simple and accurate high-level frameworks are required in order to accelerate the implementation and validation of new dynamic mapping heuristics. The proposed approach supports static (e.g. taboo search) and dynamic mapping heuristics, which were incorporated in order to support the evaluation of dynamic mapping behavior and its impact on NoC-based MPSoCs.

Due the design abstraction and flexibility we clearly see that by employing the proposed approach designers can simplify the development and the validation (e.g. more debugging capacity) of new mapping heuristics, while different scenarios can be simulated in a shorter time. For instance, to add a new mapping heuristics the designer should extend the Mapper Actor (a class) instead of modifying the HeMPS kernel.

Future works include improving the precision of the results by calibrating our unified-model using values (e.g. heuristic execution time) extracted from a real platform. In order to improve the precision of the system not only focusing on the communication but also in the processing layer, one ongoing research is developing a high-level model of PEs in order to have more information such as CPU workload and critical tasks running into the PE, that will be served as additional information for heuristics that will make decisions accordingly.

6. REFERENCES

[1] Matsutani, H.; et al. *Run-Time Power Gating of On-Chip Routers Using Look-Ahead Routing*. In: ASP-DAC'08, 2008.

[2] Marculescu, R.; et al. *Outstanding Research Problems in NoC Design: System, Microarchitecture, and Circuit Perspectives*. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 28(1), 2009.

[3] Singh, A. K.; at al. *Communication-aware heuristics for run-time task mapping on NoC-based MPSoC platforms*. Journal of Systems Architecture, 56(7), 2010.

[4] Faruque, M.A.; et al. *ADAM: Run-time Agent-based Distributed Application Mapping for on-chip Communication.* In: DAC'08, 2008.

[5] Wildermann, S.; et al. *Run time Mapping of Adaptive Applications onto Homogeneous NoC-based Reconfigurable Architectures*. In: FPT'09, 2009.

[6] Molnos, A.; et al. *Composable, energy-managed, real-time MPSOC platform*. In: OPTIM'10, 2010.

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

[8] 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.

[9] Streubühr, M. ; et al. *Efficient Approximately-Timed Performance Modeling for Architectural Exploration of MPSoCs.* In: (FDL'09), 2009.

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

[11] Hölzenspies, P.K.F.; et al. *Run-time Spatial Mapping of Streaming Applications to a Heterogeneous Multi-Processor System-on-Chip (MPSoC)*. In: DATE'08, 2008.

[12] Smit, L.T.; et al. *Run-time mapping of applications to a heterogeneous SoC.* In: SoC'05, 2005.

[13] Chou, C-L.; et al. *User-Aware Dynamic Task Allocation in Networks-on-Chip*. In: DATE'08, 2008.

[14] Carvalho, E.; et al. *Dynamic Task Mapping for MPSoCs.* Design & Test of Computers, v. 27(5), 2010.

[15] Mukherjee, R.; *Peak Temperature Control and Leakage Reduction During Binding in High Level Synthesis.* In: ISLPED'05, 2005.

[16] Coskun, A. K.; *Evaluating the Impact of Job Scheduling and Power Management on Processor Lifetime for Chip Multiprocessors.* In: SIGMETRICS'09, 2009.

[17] Mandelli, M.; *Energy-Aware Dynamic Task Mapping for NoC-based MPSoCs*. In: International Symposium on Circuits and Systems (ISCAS'11), 2011.

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

[19] Lee, E. A.; et. al. *Actor-Oriented Design of Embedded Hardware and Software.* Systems, Journal of Circuits, Systems, and Computers, 12 (3), 2003.

[20] Hu, J.; Marculescu, R. *Energy-aware mapping for tile-based NoC architectures under performance constraints.* In: Asia South Pacific Design Automation Conference (ASP-DAC'03), 2003.

[21] Ost et al. Exploring NoC-Based MPSoC Design Space with Power Estimation Models. IEEE Design and Test of Computers, 28(2), 2011.

[22] Carara, E.; et al. *HeMPS - A Framework for NoC-Based MPSoC Generation*. In: ISCAS'09, 2009.