Exploring Heterogeneous NoC-based MPSoCs: from FPGA to High-Level Modeling

Luciano Ost[1], Gabriel Marchesan Almeida[1], Marcelo Mandelli[2], Eduardo Wachter[2]; Sameer Varyani[1],
Gilles Sassatelli[1], Leandro Soares Indrusiak[3], 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, eduardo.wachter, fernando.moraes}@pucrs.br

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

Abstract— - **This paper proposes a novel strategy for enabling dynamic task mapping on heterogeneous NoC-based MPSoC architectures. The solution considers three different platforms with different area constraints and applications with distinct efficient characteristics. We propose a solution that uses a unified model-based framework, which is calibrated according to area information obtained from FPGA synthesis. Besides, we present the performance of various applications running on different processors on FPGAs aiming to obtain application efficiency characteristics for calibrating the proposed high-level model. The paper also presents three different scenarios and discusses the reduction in terms of energy consumption as well as the end-to-end communication cost for different applications such as MPEG and ADPCM, among others multimedia benchmarks.**

Keywords: modeling, NoC-based MPSoCs, design space exploration of heterogeneous MPSoCs, dynamic mapping.

I. Introduction

Heterogeneous MPSoCs are composed of different types of processing elements (PEs), dedicated IPs cores (e.g. FIR filter) that are interconnected by a network-on-chip (NoC). Heterogeneous systems provide different characteristics (e.g. floating-point and DSP operations) that can better meet the requirements of different nature of applications, providing higher performance, power efficiency and lower cost when compared to homogeneous MPSoCs [1][2][3]. Therefore, the adoption of heterogeneous properties includes new challenges to the MPSoC design flow, such as choosing a suitable platform configuration (e.g. which kind of PEs must be considered?) and organization (e.g. how distribute the PEs in the platform?), defining which heterogeneous properties (application-platform) should be considered in the mapping process.

To take the advantage of heterogeneous properties, designers should be able to explore different application-mapping-platform alternatives in order to evaluate and to optimize different performance metrics of the system (e.g. latency, throughput). Based on our own experience and on the experiments performed in this paper (Section IV.A and IV.B), we claim that to explore the challenges inherent to heterogeneous NoC-based MPSoCs using register transfer level (RTL) descriptions may be unfeasible due to some restrictions, such as:

- difficulty to integrate different PEs on the NoC-based MPSoCs platforms, since the implementation of network interfaces (NIs) requires designer knowledge (HW/SW implementation and integration, protocols, etc.) [4];
- the implementation and the analysis of runtime techniques (e.g. dynamic mapping and task migration) are PE-oriented (e.g. time to understand PE architecture is required), for instance, kernel modifications are necessary to support it [5]. In addition, the limited debugging features (kernel-oriented) difficult its evaluation, increasing the design time;
- the simulation is very time-consuming due to the number of details that are considered in the component's description. Thus, a single simulation scenario may easily take several hours, for small scenarios (e.g. 4 x 4 NoC-based MPSoCs) and it can be unfeasible for larger scenarios (days of simulation) [6].

In this context, this work contributes by proposing to use a unified model framework with the objective of exploring runtime task mapping onto heterogeneous NoC-based platforms. Processing elements are "physically" positioned according to a pre-defined area constraint obtained during a calibration phase, in an effort to accomplish a trade-off between application execution time and manufacturing cost (silicon area). Thus, custom physical layout (approximated area) can be considered in the initial design process, aiming to satisfy the performance requirements (primary goal) and the area saving (secondary goal) of the heterogeneous platform.

In order to increase the design flexibility, software designers can characterize different classes of applications, aiming to evaluate the trade-off between platform requirements support and their behavior. This work considers that a set of heterogeneous tasks can be mapped at runtime onto a set of heterogeneous PEs, according to pre-defined constraints (e.g. the computation efficiency of running a task onto a specific type of PE). Such constraints are parameterizable and can be extracted from real implementations (as demonstrated in Section IV). These constraints were included in a set of dynamic mapping heuristics in order to provide more efficient and realistic application-heterogeneous platform mapping.

This paper is organized as follows. Section II describes related works in heterogeneous MPSoCs and dynamic task mapping. Section III introduces the basic aspects of the adopted unified model-based framework, as well as a dynamic mapping heuristic used as case of study. Results, including energy consumption evaluation and application execution time are presented in Section IV. Finally, Section V points out conclusions and directions for future work.

II. State of the Art

For heterogeneous MPSoCs static mapping approaches are discussed in [7][8]. Both the papers put forward static mapping for applications such that some system characteristic is improved for example the total power consumption or the communication latency. These mappings are evaluated at design time keeping constant during the complete execution time. Static mapping are considered here as out of scope of this work. Due to the simplification of such algorithmic-based approaches (application of simple equations), they are usually faster than simulated-based RTL implementations, but with the penalty of decreasing the range of possible analyses. On the other hand, real implementations allow the detailed evaluation that produces accurate results but they are time-consuming. Examples of real heterogeneous MPSoCs platforms are described in [9][10][11]. These MPSoC are usually small (e.g. 2 RISC, 4 fixed-point DSP, 6 floating-point DSP and 3 HW IP [11]) and are developed to a specific scenario, thus static mapping are adopted as well.

Dynamic task mapping on heterogeneous MPSoC platforms are investigated in [12][13][14][15][16][17][18] [20][21]. Smit et al. [12] present an iterative hierarchical strategy to map an application to a parallel heterogeneous MPSoC architecture at run-time. Applications are modeled as a set of communicating PEs and the optimization objective is to minimize the energy consumption of the MPSoC while providing Quality of Service (QoS).

Hölzenspies et al. [13] investigate a run-time spatial mapping technique with real-time requirements, considering streaming applications mapped onto heterogeneous MPSoCs. In the proposed work, the application remapping is determined according to a set of information (i.e. latency/throughput) that is collected at design time, aiming to satisfy the QoS requirements, as well as to optimize the resources usage and minimize the energy consumption.

Faruque et al. [14] propose a distributed agent-based mapping scheme. The proposed scheme divides the system into virtual clusters. A cluster agent (CA) is responsible for all mapping operations within a cluster. Global agents (GAs) store information about all the clusters of the NoC and use a negotiating policy with CAs in order to define to which cluster an application will be mapped.

In [15] authors investigate the performance of mapping algorithms in NoC-based MPSoCs considering dynamic workloads. The heuristics targets NoC congestion minimization as a key function to optimize the NoC performance. The proposed Path Load (PL) mapping heuristic reduces the total execution time in 19.3% compared to the First Free (FF) heuristic. This work was extended in Singh et al. [16], where several communication-aware run-time multi-task mapping heuristics are proposed.

Ferrandi et al. [17] introduce an ant colony optimization (ACO) heuristic. Starting from a model of the target architecture and the application, the heuristic efficiently executes both scheduling and mapping in order to optimize the application performance. They show that compared to other approaches such as simulated annealing, tabu search, and genetic algorithms, their approach obtains better results by at least 16% on average, despite an overhead in execution time. The application scenario is based on JPEG encoder and the approach is validated on realistic target architecture.

Huang et al. [18] present a novel technique that is able to minimize the energy consumption of the entire multi-mode system while satisfying a given lifetime reliability constraint. Experimental results are conducted on several hypothetical MPSoC platforms with various applications modeled as task graphs. By using both single-mode and multi-mode approaches, their technique provides up to 26.3% and 27.8% energy reduction when compared with greedy solutions [19], respectively.

Chen et al. [20] investigate a method that explores how to synthesize a heterogeneous multiprocessor platform or select processing units with the partitioning of real-time tasks so that the energy consumption is minimized. By considering both static (leakage) power consumption and dynamic power consumption, they propose a method that uses a polynomial-time approximation algorithm to minimize the energy consumption by applying the minimum-average-energy-first strategy.

Kim et al. [21] introduce a cosynthesis framework for design space exploration that considers heterogeneous scheduling while mapping multimedia applications onto such MPSoCs. The optimization key is energy, so that a suitable scheduling policy is chosen for each IP in the architecture. Experimental results on a realistic multimode multimedia terminal application demonstrate that the proposed approach enables at selecting design points with up to 60.5% reduced energy for a given area constraint.

Table I summarizes the reviewed works. To the best of our knowledge, we are presenting the first scheme for enabling dynamic task mapping in heterogeneous MPSoCs architectures that considers real area occupation in FPGA as calibration information to be used in a unified model based framework. Besides, we extended a dynamic mapping heuristic that consider both area and application efficiency constraints, obtained from real implementations on FPGAs running onto different processors. Bringing all together into a unified model based framework we are able to analyze different behaviors and efficiency of the adopted heuristic given certain constraints. We also evaluate the energy consumption considering three scenarios with different area and application efficiency constraints.

Table I RELATED WORKS ON TASK MAPPING ON HETEROGENEOUS MPSOCS ARCHITECTURES

Reference	***Heuristics***	***Mapping Approach***	***Optimization Criterion***	***Abstraction Level***
Hu et al. [7]	(i) Greedy incremental algorithm	Static mapping	Energy consumption	Algorithmic
Marcon et al. [8]	(i) Largest communication first (LCF) , (ii) Greedy incremental (GI), (iii) taboo search (TS), (iv) simulated annealing (SA)	Static mapping	Energy consumption	Algorithmic
Smit et al. [12]	(i) Hierarchical iterative approach	Dynamic mapping	Energy consumption	Algorithmic
Hölzenspies et al. [13]	(i) Hierarchical search with iterative refinement	Dynamic mapping	Energy consumption	Algorithmic
Faruque et al. [14]	(i) Run-time application mapping in a distributed manner using agents	Dynamic mapping	Execution time, mapping time	Algorithmic
Carvalho et al. [15]	(i) First Free (FF), (ii) Next Neighbor (NN), (iii) Minimum Maximum Channel Load (MMC), (iv) Minimum Average Channel Load (MAC) and (v) Path Load (PL)	Dynamic mapping	Network contention, communication volume	TLM
Singh et al. [16]	(i) Communication-aware nearest neighbor (CNN), (ii) Communication-aware packing-based nearest neighbor (CPNN), (iii) Communication-aware best neighbor (CBN) and (iv) Communication-aware packing-based best neighbor (CPBN)	Dynamic mapping	Network contention, communication volume and energy consumption	TLM
Ferrandi et al. [17]	(i) Ant Colony Optimization (ACO)	Dynamic mapping	Application execution time	RTL
Huang et al. [18]	(i) Task allocation and scheduling under lifetime reliability constraint	Dynamic mapping	Energy consumption	Algorithmic
Chen et al. [20]	(i) Algorithm MAEF (Minimum-average-energy-first)	Dynamic mapping	Energy consumption	Algorithmic
Kim et al. [21]	(i) Multiway KLFM (Kernighan–Lin/Fiduccia–Mattheyses)	Dynamic mapping	Energy consumption	Algorithmic
Proposed approach	(i) Lower Energy Consumption based on Dependencies-neighborhood (LEC-DN)	Dynamic mapping	Energy consumption, hotspots and application execution time	RTL (calibration) and TLM (simulation)

III. UNIFIED MODEL-BASED FRAMEWORK

This section presents a unified model-based framework, which is organized in three main layers, as illustrated in Fig. 1: (*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).

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. Thus, the three layers are independent but the proposed approach supports the joint validation of applications mapped onto the platform model to establish a good trade-off between the platform characteristics and the requirements of the given application.

With this framework, both heterogeneous advantage and dynamic mapping benefits can be easily evaluated due to the modeling and design flexibility of proposed approach. In this context, system designers can take advantage of application-platform heterogeneous properties when mapping tasks to specific processor types (e.g. MIPS, DSP, *Platform Layer* in Fig. 1), while optimizing performance metrics such as latency, area cost and energy consumption.

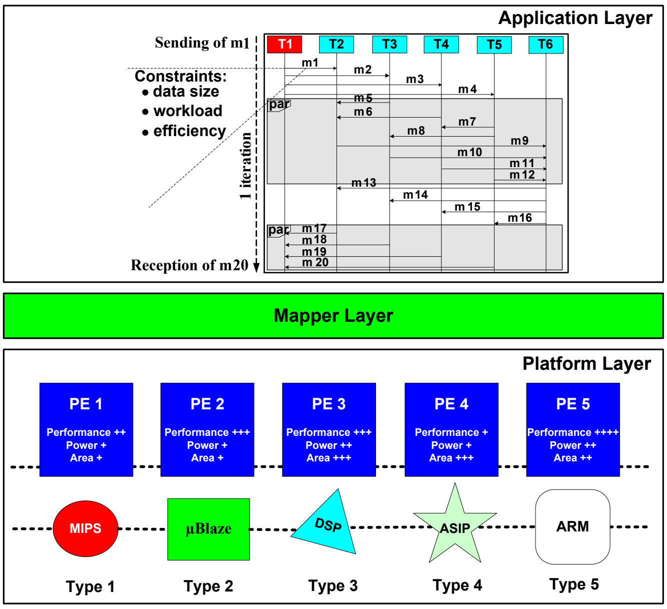


Fig. 1 - Heterogeneous architecture illustrating particular characteristic of each IP core (e.g. PEs).

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

A. *Application Layer*

Most researchers on MPSoCs 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 to the software engineer 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 [23].

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 orientation 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 [24]. 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.

In this scenario, the *Application Layer* was extended in order to support application characteristics, called here as heterogeneous properties (e.g. efficiency), which are considered in mapping decision strategies (either initial mapping or run-time mapping). Thus, during the mapping process such application properties are considered as cost functions, leading to higher overall system performance, since the mapper tries to fit these properties to the PE characteristics. For instance, the user can define that a task can be executed in more than one type of PE by configuring the *task type*. Each *task type* can be executed onto a certain PE, which differs in terms of task execution efficiency (e.g. efficiency parameter, *Application Layer* in Fig. 1). Such parameters can be calibrated from a real implementation, as shown in Section IV.A.

B. *Platform Layer*

The present approach provides a set of NoC 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. Abstract model of PEs and scheduling algorithms (e.g. First-Come-First-Served, FCFS – adopted in the present work) are also provided, allowing more realist traffic generation (e.g. ON-phase and OFF- phase period).

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 [7] and/or rate-based [23] NoC power models.

C. *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, simulated annealing [23]) and dynamic mapping heuristics (e.g. LEC-DN [5], adopted in this work). The LEC-DN was chosen to be extended to consider heterogeneous properties as cost functions due to the following reasons: (*i*) it was implemented and validated in a RTL homogeneous NoC-based platform; (*ii*) results show a reduction in term of execution time (4.3% in average) and the energy consumption (10.8% in average) [5], when compared to well-known dynamic mapping heuristics [15].

The LEC-DN employs two cost functions: (*i*) proximity, in number of hops; (*ii*) communication volume between 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) [15]. In addition to the two previous cost functions, the heterogeneous LEC-DN considers task execution efficiency before mapping a required task onto a PE. 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.

Consider the application illustrated in Fig. 2(a), containing 4 tasks, where AB_1 and AB_2 are initial tasks.

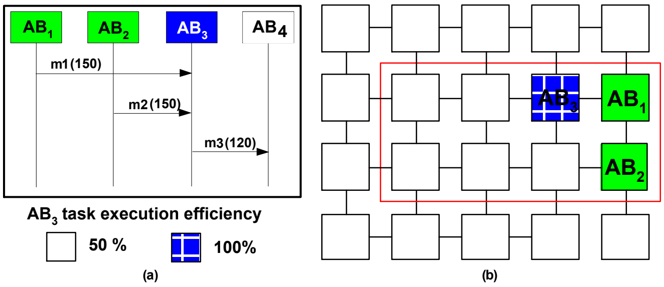


Fig. 2 - (a) application described as UML diagram; (b) search space to map the AB_3 task, and one possible mapping for AB_3.

[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.

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 (Fig. 2 (b)). Thus, AB_3 will be mapped nearest to task AB_1, since according to the application UML the task execution efficiency is higher in the PE next to AB_1 than in PE next to AB_2 (50% normalized value based on higher task execution efficiency, in case, 100%). Note that task AB_4 is not mapped, since it depends from task AB_3.

IV. STUDY CASES AND RESULTS

A. *Application Performance*

In our experiments, we have used a hardware prototype called XScale 270 [23], which is based on ARMv5 processor architecture. The ARMv5 is the master node responsible for system monitoring and application scheduling. The processor is connected to Spartan3-5000 FPGA in which we have a dual processor µBlaze implementation with both µBlaze processors configured differently. The communication between the XScale and the µBlaze processor is done by using a bridge directly connected to the PLB bus of one of the µBlaze processors. Each µBlaze processor is configured to have a small local memory and communication between them is allowed by using mailboxes (message passing). For the purpose of heterogeneity each processor is configured differently based on the presence or absence of hardware multiplier and barrel shifter, as presented on Table II.

Table II DIFFERENT CONFIGURATIONS OF µBLAZE PROCESSORS

	Hardware Multiplier	*Barrel Shifter*	*#LUTS*
µBlaze1	Yes	Yes	3631
µBlaze2	Yes	No	3404
µBlaze3	No	Yes	3581
µBlaze4	No	No	3357

Fig. 3 depicts the performance of different benchmarks applications. Applications performance is evaluated in terms of required number of clock cycles for application execution. We present four different configurations of µBlaze processors (as presented in Table II) and two different configurations of the ARM processor: (i) ARM 520 running at 520 MHz and (ii) ARM 208 running at 208 MHz, both running a flavor Linux kernel. Results show that each application may perform better on a particular target depending on both application and processor characteristics. It is possible to observe that the performance of FIR application is dependent on the presence of a hardware multiplier whereas inverse quantization can benefit equally from both multiplier and barrel shifter. Another interesting point to be observed is that FIR and Iquant applications perform better on µBlaze1 and µBlaze2 processors since it is simple processor with the hardware multiplier with only 5 pipeline stages and without MMU and Cache, which makes this processor more efficient than ARM.

Since applications can be executed on different processors, they may perform differently and in most cases µBlaze processor performs as good as XScale processor. This principle is used to map the application on given processing element depending on the requirements.

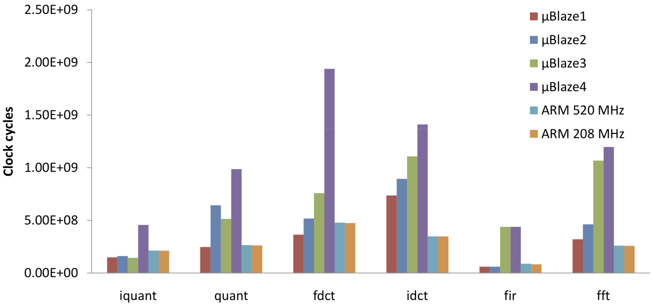


Fig. 3 - Normalized performance of benchmarks running on different processing nodes.

B. *Area Overhead*

Kranenburg [22] evaluates open source processors, considering design quality, performance, available documentation and the set of specific features of each one. In addition to the Kranenburg approach [22], the Plasma Processor, COFFEE processor, and Plasma with Floating Point Unit were evaluated in terms of resource utilization. Fig. 4 presents the area occupation in terms of flip-flops and four-input LUTs for eight different processors.

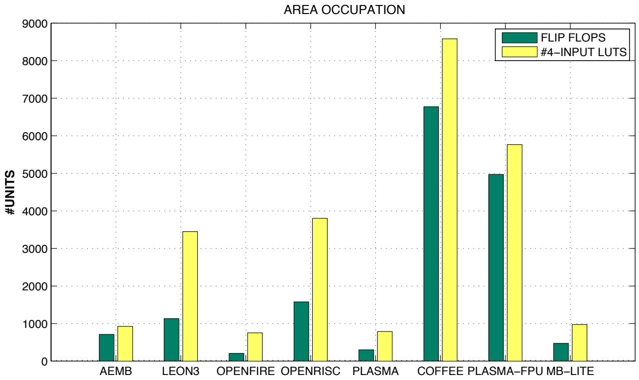


Fig. 4 - Resource utilization (flip-flops - FFs and Look Up Tables - LUTs) of evaluated processors, targeting the xc5vlx110ff1760-3 device, except Plasma-FPU. Adapted from [22].

Fig. 5 ilustrates the area occuppation for both Plasma and MB-Lite processors. The PE is composed of a NoC (Router), a DMA controller, a Network Interface (NI) and peripherals (Others). It is possible to observe that the processor is responsible for 80% of the processing element area. For this reason we have only considered different processors in our high-level model, keeping the same router for all of them.

C. *High-Level Modeling*

For the sake of simplicity, three processors type were employed: MB-Lite, LEON and the Plasma-FPU (considered here as dedicated IP core). Table III describes architectural characteristics of the modeled heterogeneous MPSoC system.

Table III ARCHITECTURAL CHARACTERISTICS OF OUR HETEROGENEOUS MPSOC SYSTEM

NoC Dimension	*Number of Clusters*	*Number of PEs per cluster*	*Total PEs*	*MB-Lite (#LUTS)*	*Leon (#LUTS)*	*Plasma-FPU (#LUTS)*
7x7	4	12	49	1,000	2,700	5,800

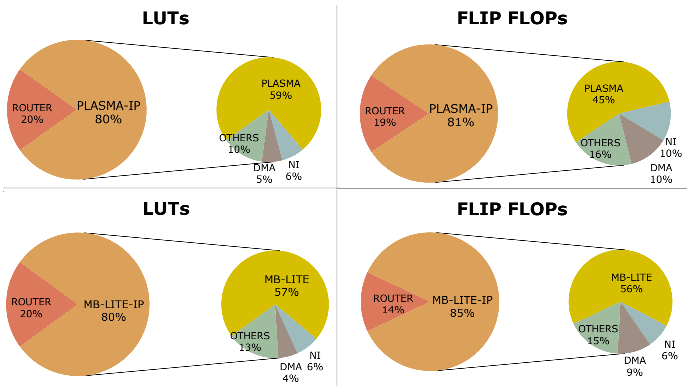


Fig. 5 - Area occupation for both Plasma and MB-Lite processors.

The architecture is split into four clusters, each one containing twelve processing elements together with a central node (Mapper) responsible for dynamically mapping tasks onto PEs. Three different processors are modeled from real area occupation values obtained from synthesis process on FPGA. It is possible to observe that MB-Lite processor occupies only 1,000 LUTs while Plasma-FPU [25] (Plasma with floating point unit) is almost six times bigger, occupying 5,800 LUTs.

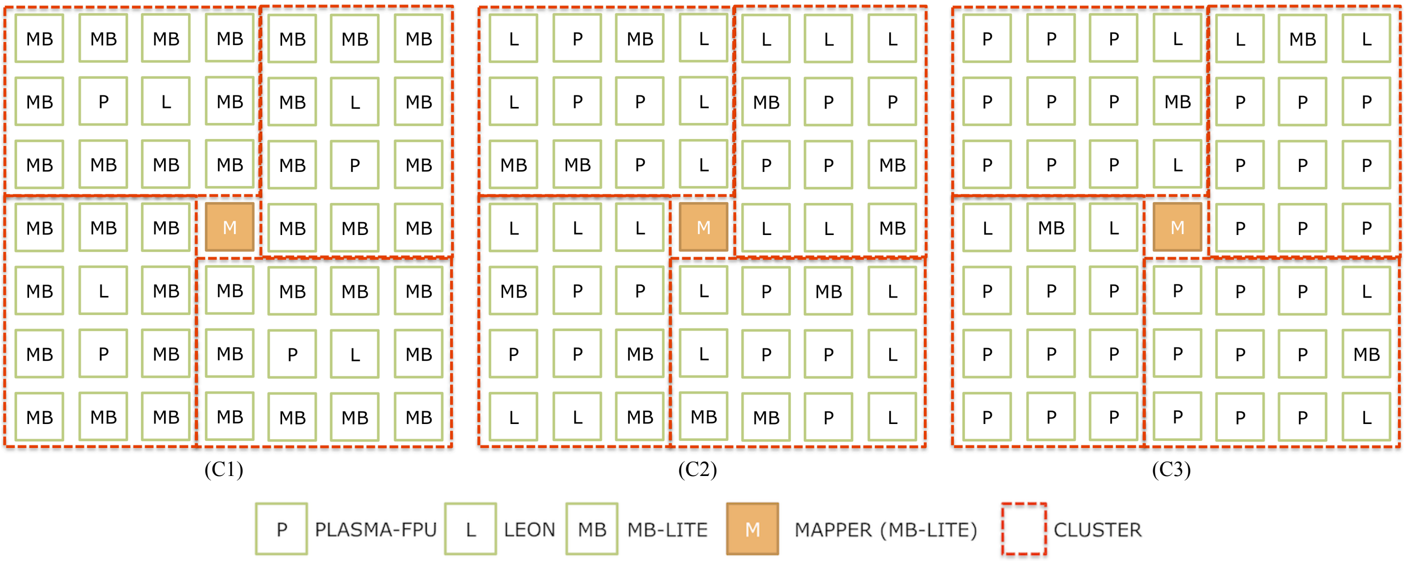
After having obtained area occupation figures for different processors, three hyphotetical configurations were created according to area constraints. For each configuration there is an available area per cluster where twelve processors must fit on it.

1. Configuration 1 (C1): the architecture is built of ten MB-Lite processors together with 1 Leon and 1 Plasma-FPU processors. The resulting occupied area is 18,500 LUTs;
2. Configuration 2 (C2): the architecture is built of three MB-Lite processors together with 5 Leon and 4 Plasma-FPU processors. The resulting occupied area is 39,700 LUTs;
3. Configuration 3 (C3): the architecture is built of one MB-Lite processor together with 2 Leon and 9 Plasma-FPU processors. The resulting occupied area is 58,600 LUTs (Table IV);

Table IV PROCESSORS CONFIGURATION FOR THREE DIFFERENT SCENARIOS

Configu-ration	*Available area per cluster (#LUTS)*	*Cluster$_i$*			*Occupied area per cluster (#LUTS)*
		MB	L	P	
C1	20,000	10	1	1	18,500
C2	40,000	3	5	4	39,700
C3	60,000	1	2	9	58,600

Fig. 6 illustrates three heterogeneous platforms considering the different configurations previously presented. The central node (M) is a MB-Lite based processor which hosts the *Mapper Actor*, responsible for dynamically mapping tasks into different PEs in the architecture.



MB	MB	MB	MB	MB	MB	MB
MB	P	L	MB	MB	L	MB
MB	MB	MB	MB	MB	P	MB
MB	MB	MB	M	MB	MB	MB
MB	L	MB	MB	MB	MB	MB
MB	P	MB	MB	P	L	MB
MB	MB	MB	MB	MB	MB	MB

(C1)

L	P	MB	L	L	L	L
L	P	P	L	MB	P	P
MB	MB	P	L	P	P	MB
L	L	L	M	L	L	MB
MB	P	P	L	P	MB	L
P	P	MB	L	P	P	L
L	L	MB	MB	MB	P	L

(C2)

P	P	P	L	L	MB	L
P	P	P	MB	P	P	P
P	P	P	L	P	P	P
L	MB	L	M	P	P	P
P	P	P	P	P	P	L
P	P	P	P	P	P	MB
P	P	P	P	P	P	L

(C3)

Fig. 6 - Representation of three heterogeneous platforms considering different area constraints (available area per cluster): C1) 20,000 LUTs; C2) 40,000 LUTs; C3) 60,000 LUTs.

Table V END-TO-END COMMUNICATION COST REPRESENTATION CONSIDERING THREE SCENARIOS WITH DIFFERENT AREA CONSTRAINTS AND FIVE APPLICATIONS

Scenario	S1		S2		S3	
Area (#LUTS)	20,000		40,000		60,000	
Energy VB (µJ)	32.89		32.89		32.89	
Applications	MAX($EEC_{i,j}$)	AV($EEC_{i,j}$, N)	MAX($EEC_{i,j}$)	AV($EEC_{i,j}$, N)	MAX($EEC_{i,j}$)	AV($EEC_{i,j}$, N)
MPEG	885,434	883,904	615,230	613,486	441,870	440,659
MWD	101,006	100,349	62,618	62,265	42,102	41,469
SEGIMG	1,012,450	801,152	699,538	545,951	348,824	279,112
SYNTHETIC	282,709	278,038	181,173	179,018	146,009	142,837
AAV	1,006,060	782,409	851,484	627,100	487,326	390,101
Total	3,287,659	2,845,852	2,410,043	2,027,820	1,466,131	1,294,178

Table VI END-TO-END COMMUNICATION COST REPRESENTATION CONSIDERING THREE SCENARIOS WITH DIFFERENT AREA CONSTRAINTS AND FIVE APPLICATIONS WITH DIFFERENT EFFICIENCY CHARACTERISTICS

Scenario	S1		S2		S3	
Area (#LUTS)	20,000		40,000		60,000	
Energy VB (µJ)	30.08		33.05		32.89	
Applications	MAX($EEC_{i,j}$)	AV($EEC_{i,j}$, N)	MAX($EEC_{i,j}$)	AV($EEC_{i,j}$, N)	MAX($EEC_{i,j}$)	AV($EEC_{i,j}$, N)
MPEG	885,434	883,904	550,631	550,187	441,870	440,659
MWD	99,740	99,069	69,650	69,310	40,818	40,818
SEGIMG	816,302	796,370	647,721	514,071	348,824	279,112
SYNTHETIC	252,228	247,545	156,961	153,116	118,431	116,185
AAV	1,010,440	786,815	788,016	564,357	487,326	390,101
Total	3,064,144	2,813,703	2,212,979	1,851,041	1,437,269	1,266,875

Note that the all tasks execution efficiency were defined as 20% for MB-Lite, 50% for LEON and 100% for Plasma-FPU. Table V presents the end-to-end communication cost considering three scenarios with different area constratints and five applications. For each scenario we have calculated the end-to-end cost which is given by Equation 1.

$$\mathrm{EEC}_{i,j} = \mathrm{P}_i + \mathrm{L}_{i,j} \qquad (1)$$

where,

$\mathrm{EEC}_{i,j}$ = End-to-end communication cost between source node (task *i*) and destination node (task *j*);

P_i = Processing time of task *i;*

$\mathrm{L}_{i,j}$ = Latency between source node (task *i*) and destination node (task *j*);

Besides, a $MAX(EEC_{i,j})$ value is calculated based on one single frame execution while $AV(EEC_{i,j}\ ,\ N)$ represents the average value for *N* inter-task communication events of a set of applications.

It is possible to observe that as available area is increased, the end-to-end communication cost is reduced and the same behavior is noted for all five applications. This is explained by the fact that as available area increases (S3), it is possible to fit more powerful processors units such as Plasma-FPU into the cluster. As a result the processing cost and the $\mathrm{EEC}_{i,j}$ are respectively reduced. If we consider the MPEG application, we can observe that the average $AV(EEC_{i,j}, N)$ is reduced in 31% and 50% when comparing S1 against S2 and S3 respectively. For SEGIMG application, the $EEC_{i,j}$ reduction is of 31% and 72% respectively. Considering the total average comprising the five applications, we can clearly see that there is a reduction of 29% (S2) and 55% (S3) compared to S1.

The energy consumption of the NoC remains the same in all scenarios, since the communicaton through the NoC is not considelable (less than 12% of the available throughput) and only one mapping heuristic is employed.

Table VI presents the end-to-end communication cost considering different area constraints and five applications with distinct efficiency characteristics. As explained in Section III. A, applications are modeled with different efficiency characteristics for different processors. Heach one of the three different processors (MB-Lite, LEON and Plasma-FPU), where it is possible to observe that compared to Table V, which application efficiency is not considered, this approach performs better in all three scenarios. Concerning energy consumption, S3 performs slightly better compared to S1 and S2 (30.08µJ and 33.05µJ respectively).

V. CONCLUSIONS

High-level models are concrete solutions that are mainly powerful due to their capacity of representing real architectures and validating scenarios that would might not be possible to be explored in a feasible time on, i.e., RTL-based architectures. On the other hand, the calibration of such models must be done very carefully and the information to fed the model should have a high degree of precision.

In this paper we have proposed a novel solution that uses a unified model-based framework in which different heterogeneous architectures are modeled and validated. The model is calibrated with synthesis information of different embedded processors obtained from real implementation on FPGA. Different applications run on each processor and distinct performance figures area obtained. Thus, we propose a novel solution for representing application efficiency characteristics according to different processors. The paper presents three scenarios that consider clusters with different area constraints and multimedia applications such as MPEG and ADPCM are used as benchmarks for the case studies.

We compare energy consumption efficiency of three different architectures as well as the end-to-end communication cost for all five applications. We can clearly observe that dynamic task mapping performs better when considering application efficiency before mapping tasks into processing elements. As future work we plan to also consider the energy consumption of processors in order to improve the decision-making when tasks are dynamically mapped onto the architecture.

REFERENCES

[1] Sharifi, S.; et al. “Hybrid Dynamic Energy and Thermal Management in Heterogeneous Embedded Multiprocessor SoCs”**.** In: Asia South Pacific Design Automation Conference (ASP-DAC'10), 2010.

[2] Jerraya, A.; A.; Wolf, W. “*Multiprocessor Systems-on-Chips*”. Morgan Kaufmann, 2005, 602p.Leupers, R. et al. “Cool MPSoC programming”. In: (DATE'10), 2010**.**

[3] Shen, H. "Novel Task Migration Framework on Configurable Heterogeneous MPSoC Platforms". In: Asia South Pacific Design Automation Conference (ASP-DAC'09), 2009.

[4] Grasset, A.; “Network interface generation for MPSOC: from communication service requirements to RTL implementation”. In: Rapid System Prototyping (RSP'04), 2004.

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

[6] Roth, C.; et al. “Modular Framework for Multi-level Multi-device MPSoC Simulation”. In: Reconfigurable Architectures Workshop (RAW'11), 2011.

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

[8] Marcon, C.; et al. “Comparison of network-on-chip mapping algorithms targeting low energy consumption”. IET Computers and Digital Techniques, vol. 2(6), 2008.

[9] Zhang, W; et al. “Design of heterogeneous MPSoC on FPGA”. In: International Conference on ASIC (ASICON’07), 2007.

[10] Jalier, C.; et al. “Heterogeneous vs Homogeneous MPSoC Approaches for a Mobile LTE Modem”. In: Design, Automation & Test in Europe Conference & Exhibition, 2010.

[11] Limberg T.; et. al. “A Heterogeneous MPSoC with Hardware Supported Dynamic Task Scheduling for Software Defined Radio”. In: Design Automation Conference (DAC'09), 2009.

[12] Smit, L.T.; et al. “Run-time mapping of applications to a heterogeneous SoC”. In: International Symposium on System-on-Chip (SoC'05), 2005.

[13] Hölzenspies, P.K.F.; et al. “Run-time Spatial Mapping of Streaming Applications to a Heterogeneous Multi-Processor System-on-Chip (MPSoC)”. In: Design, Automation and Test in Europe (DATE'08), 2008.

[14] Faruque, M.A.; et al. “ADAM: Run-time Agent-based Distributed Application Mapping for on-chip Communication”. In: Design Automation Conference (DAC'08), 2008.

[15] Carvalho, E.; et al., “Heuristics for Dynamic Task Mapping in NoC-based Heterogeneous MPSoCs”. In: International Workshop on Rapid System Prototyping (RSP'07), 2007.

[16] 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.

[17] Ferrandi, F.; et al. “Ant colony heuristic for mapping and scheduling tasks and communications on heterogeneous embedded systems”. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 29(6), 2010.

[18] Huang. L.; et al. “Energy-efficient task allocation and scheduling for multi-mode MPSoCs under lifetime reliability constraint”. In: Design, Automation and Test in Europe (DATE'10), 2010.

[19] Bjorn-Jorgensen and J. Madsen, “Critical Path Driven Cosynthesis for Heterogeneous Target Architectures”. In: International Conference on Hardware⁄Software Codesign (CODES’97).

[20] Chen, J. J.; et al. “Platform synthesis and partitioning of real-time tasks for energy efficiency”. Journal of Systems Architecture, ISSN: 1383-7621, 2010.

[21] Kim, M.; et al. “Energy-aware cosynthesis of real-time multimedia applications on MPSoCs using heterogeneous scheduling policies”, ACM Transactions on Embeded Computer Systems, vol. 7(2), 2008.

[22] Kranenburg,T.; “Reference design of a portable and customizable microprocessor for rapid system prototyping,” Master’s thesis, Delft University, 2009.

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

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

[25] Rodolfo, T.A.; et al. "Floating Point Hardware for Embedded Processors in FPGAs: Design Space Exploration for Performance and Area". In: International Conference on Reconfigurable Computing and FPGAs (ReConFig’09), 2009.