Partitioning and Mapping on NoC-Based MPSoC: An Energy Consumption Saving Approach

Eduardo Antunes, Alexandra Aguiar, Sergio Johann F., Marcos Sartori, Fabiano Hessel, César Marcon
PPGCC - Post-Graduation Program in Computer Science
PUCRS - Pontifical University Catholic of Rio Grande do Sul
+55 (51) 3320-3558, Porto Alegre, Brazil
{eduardo.brum, alexandra.aguiar, sergio.filho, marcos.sartori, fabiano.hessel, cesar.marcon}@pucrs.br

ABSTRACT
Software complexity has increased considerably over recent years, needing special target architectures as MPSoCs to fulfill the heavy memory, communication and computation requirements. Nevertheless, the use of MPSoCs has brought attention to the need for effective methods and tools for parallel software development. Methodologies aggregating partitioning and mapping are normally employed to fulfill the heavy requirements of such systems. This paper explores task-partitioning and processor-mapping methods on homogeneous NoC-Based MPSoC. The effect of both on application's energy consumption is explored alone and jointly. Experiments with several synthetic and four real applications show that the energy consumption is reduced up to 18%, 31.8% or 38.1% when applying partitioning, mapping or both, respectively.

Categories and Subject Descriptors
B.7 **[Integrated Circuits]:** Advanced technologies, VLSI

General Terms
Performance, Design

Keywords
Partitioning, Mapping, MPSoC, NoC

1. INTRODUCTION
Recent years have brought a large quantity of application, demanding huge computational power, reduced energy consumption and efficient communication, which boast the research and development of special target architectures, like a NoC-based MPSoC. This one implements the complete system functionality into a single chip and support the heavy communication requirements of hundreds cores.

From the processing point of view, homogeneous MPSoCs are those composed by processors of the same type and heterogeneous MPSoCs are those composed by at least two processors with different architectures. Heterogeneous MPSoCs can support a wide variety of applications, since each processor has specific computation and communication features. Otherwise, homogeneous MPSoCs are easier to program, increase the mapping and partitioning possibilities, and enable global load balancing through application-task migration. Besides, the homogeneity may reduce the global energy consumption and area occupation for some set of applications [1].

NoCArc '11, December 4, 2011, Porto Alegre, Brazil

The development of efficient application for homogeneous MPSoCs is a current challenge, especially regarding to the workload balancing among processors. Therefore, this work considers a homogeneous NoC-based MPSoC as target architecture, and presents a partial design flow containing the application-task partitioning into groups of tasks, where each group is associated to a single processor, and the processors mapping onto tiles of the NoC architecture. Whereas tile is a limited area of the target architecture, comprising a processor, a router, a local memory and auxiliary circuits.

Moreover, we demonstrate the effect of partitioning and mapping alone and combined on the energy consumption saving. Experimental results show that an MPSoC may save significant energy when applying only partitioning or mapping. Furthermore, when combining both activities, the energy consumption is reduced up to 38.1%.

Several works relate to processor mapping onto NoC-based architectures and some ones describe the tasks partitioning into groups associated to processors ([2], [3], [4], [5], [6], [7], [8] and [9]), **but none compare the joint effect of both**. A partial comparison of these works is summarized in Table 1. Moreover, a number of works uses the same name "mapping" to define both mapping and partitioning, while the name "partitioning" is used only to explore hardware/software division.

Table 1 - Related work comparison.

Work	MPSoC type	Target Architecture	Activity / Dynamic	Objective
[2]	Heterogeneous	NoC	Partitioning and mapping / static	Execution time and area reduction
[3]	Heterogeneous	NoC	Partitioning and mapping / static	Latency minimization and throughput maximization
[4]	Heterogeneous	Generic	Partitioning and mapping / static	Load balancing
[5]	Homogeneous	NoC	Partitioning and mapping / static	Execution time and power consumption reduction
[6]	Homogeneous	NoC	Partitioning and mapping / static	Communication latency reduction
[7]	Homogeneous	Bus	Scheduling and Partitioning / static	Execution time reduction
[8]	Homogeneous	Generic	Partitioning and mapping / static and dynamic	Execution time reduction
[9]	Heterogeneous	Generic	Partitioning / static	Execution time reduction
This work	**Homogeneous**	**NoC**	**Partitioning and mapping / static**	**Partitioning versus mapping exploration (Energy consumption)**

This paper is organized in six sections. Section 2 presents the partitioning and the mapping problem formulation together with the underlying data structures and the energy model. Section 3 describes the methodology and the tools used to accomplish the experimental results. Section 4 employs an application to exemplify the methodology. Section 5 shows experimental results and Section 6 concludes the paper.

2. PROBLEM FORMULATION

The complete homogeneous MPSoC design implies several activities with some specificity according to the application description nature and the target architecture. Here, we describe two design activities, which are the task partitioning and the processor mapping.

Parallel applications are described as a set of communicating tasks. According to some requirements (e.g. energy consumption reduction) and some constraints (e.g. number of target processors) these tasks are grouped. The grouping of all application tasks, which is the *task partitioning* activity, generates a partition.

Since all processors of homogeneous MPSoC are the same type, the *binding* activity may be easily performed by assigning each group to a single processor.

Once communicating processors represent the application, the next step is to perform the selection of the best place to insert each one of these processors, which is the *processor mapping* onto tiles activity. According to the target architecture, the mapping may severely affect some design requirements, such as latency minimization and energy consumption saving.

To better understand partitioning and mapping concepts, Figure 1 exemplifies the partitioning of a hypothetical application composed by 22 tasks into 6 groups that are associated to 6 processors and the corresponding mapping of these processors onto a 2D-mesh NoC architecture. The application is composed by a set of parallel communicating tasks $T = \{t_1, t_2, \ldots, t_{22}\}$. The tasks partitioning, which is represented by continuous arrows, generates $G = \{g_1, g_2, \ldots, g_6\}$ that is a set of groups of tasks.

Subsequently, the binding activity, which is represented by dashed arrows, associate each element of G to a single processor of the set of processors $P = \{p_1, p_2, \ldots, p_6\}$. Finally, the processor mapping onto the set of NoC tiles $\Gamma = \{\tau_1, \tau_2, \ldots, \tau_6\}$ is represented by the dotted arrows.

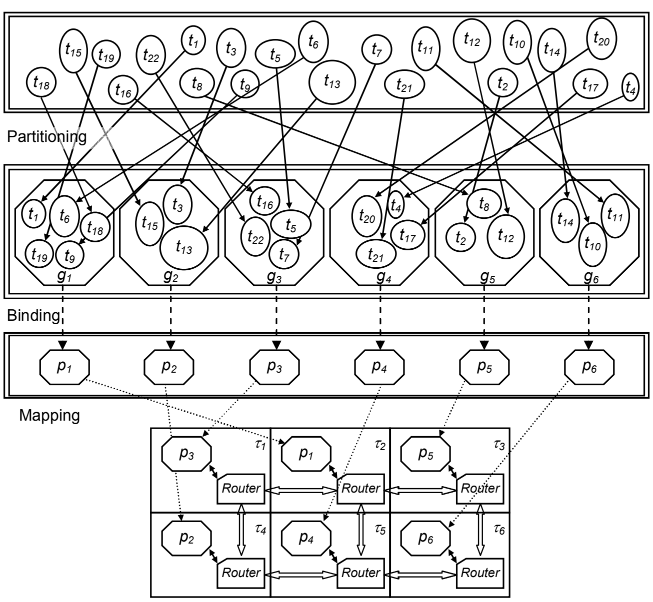


Figure 1 – Task partitioning and processor mapping.

2.1 Partitioning and Mapping Complexities

The design flow presented here uses an approach that separates the steps of partitioning tasks across groups, binding these groups into processors and mapping processors into tiles. It is important to highlight that several works map the application tasks directly onto the target architecture tiles, excluding binding and partitioning. However, due to the NP-complete nature of these activities [10], the results obtained with approaches that map application tasks directly to the target architecture tend to be worse when compared to our approach, mainly in the cases where the task is done at run time, since the mapping has a short time to be accomplished.

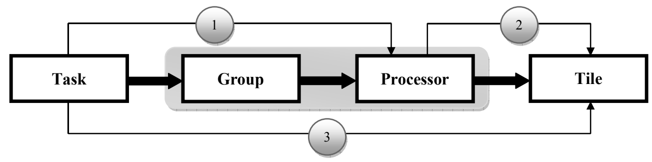


Figure 2 – The flow containing each step from the partitioning of tasks into groups, the binding groups into processors and the processors mapping onto tiles of the target architecture.

The partitioning of tasks into groups (arrow 1 in Figure 2) is an activity with complexity proportional to the Bell number $O(Bell(n))$ [11], where n is the number of tasks, since there is no order relation between groups and even within a group.

The mapping processor (composed of tasks groups) in tiles of the target architecture (arrow 2 of Figure 2) is $O(t!)$ complex, where t is the number of tiles, because it reflects all combinations of positions of processors in all tiles.

The mapping of tasks to tiles of the target architecture (arrow 3 of Figure 2) adds the complexities of partitioning tasks across groups, binding each to a processor and mapping of the processors in tiles of the target architecture. In this case, the complexity is much higher $O(Bell(n) \square t!)$.

Figure 3 shows that the number of solutions to be explored with mapping tasks onto tiles is much higher than the others. Thus, even applying good algorithms, the results obtained with this activity tend to be worse, when compared to those obtained with the flow proposed here.

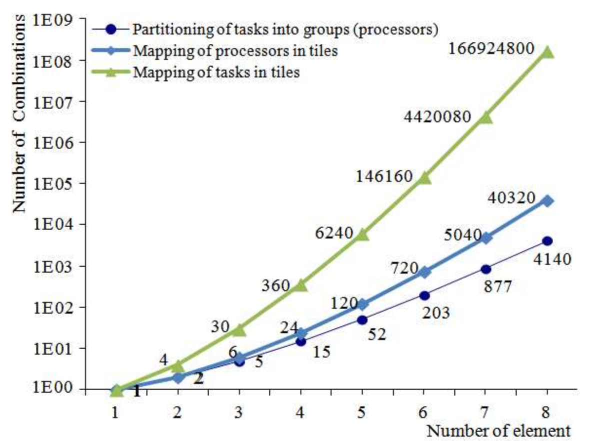


Figure 3 – Number of combinations against the number of elements (logarithmic scale): (i) Partitioning of tasks into groups (i.e. processors), (ii) Mapping of processors in tiles, and (iii) Mapping of tasks onto tiles.

2.2 Structures Definitions

The partitioning and mapping have three main data structures that are set out below.

Definition 1: A *Task Communication Graph* (TCG) is a directed graph $<T, V>$. The set of vertices $T = \{t_1, t_2, \ldots, t_m\}$ represents the set of m tasks in one parallel application. Assuming v_{ab} is the bits amount of all packets sent from a task t_a to a task t_b, then the set of edges V is $\{(t_a, t_b) \mid t_a, t_b \in T$ and $v_{ab} \neq 0\}$, and each edge is labeled with the value v_{ab}. V represents all communications between the application tasks.

Definition 2: A *Communication Weighted Graph* (CWG) is a directed graph $<P, W>$, similar to the TCG. However, the set of vertices $P = \{p_1, p_2, \ldots, p_n\}$ represents the set of processors in one application. The number of processors n is equal to the total number of tiles. Furthermore, w_{ab} is the total number of bits sent from a processor p_a to a processor p_b. Then the set of edges W is $\{(p_a, p_b) \mid p_a, p_b \in P$ and $w_{ab} \neq 0\}$, and each edge is labeled with the value w_{ab}. W represents all communications between the MPSoC processors, while CWG reveals information of application's relative communication volume.

The processor mapping is performed regarding to a 2D mesh NoC using wormhole and deterministic XY routing algorithm. The communication resource graph stated below captures the NoC topology.

Definition 3: A *Communication Resource Graph* (CRG) is a directed graph $<\Gamma, L>$, where the vertex set is the set of tiles $\Gamma = \{\tau_1, \tau_2, \ldots, \tau_n\}$ and the edge set $L = \{(\tau_i, \tau_j), \forall\ \tau_i, \tau_j \in \Gamma\}$ gives the set of paths from τ_i to τ_j. The value n is again the total number of tiles and is equal to the product of NoC lines and columns. CRG edges and vertices represent physical links and routers of the target architecture, respectively. The CRG definition is equivalent to the architecture characterization graph in [12] and to the NoC topology graph in [13].

2.3 Energy Model

Both, processors (with the whole memory hierarchy) and communication architecture originate energy consumption.

The sum of the energy consumed by the execution of all tasks grouped on a processor enable estimating its energy consumption. This value is used, together with the communication volume between tasks, to choose good partitions. On the other hand, the amount of bits transmitted between tasks grouped and mapped into different processors contributes to estimate the energy consumption used to choose good mappings.

The approach used here to model the NoC's energy consumption is similar to those shown in [12] and [14]. Dynamic energy consumption is proportional to switching activity, arising from packets moving across the NoC, dissipating energy on the links and inside of each router. The concept of bit energy *EBit* [14] is used to estimate the dynamic energy consumption of each bit, when this flips its polarity from a previous value. *EBit* is split into three components: (i) bit dynamic energy consumed by the router (wires, buffers and logic gates) (*ERbit*); (ii) bit dynamic energy consumed on horizontal (*ELHbit*) and vertical (*ELVbit*) links between tiles; and (iii) bit dynamic energy consumed on the links between the router and the local processor (*ECbit*). Equation (1) expresses the relationship between these quantities, which computes the dynamic energy consumption of a bit passing through a router, a vertical or horizontal link and a local link.

$$(1) \quad EBit = ERbit + (ELHbit \text{ or } ELVbit) + ECbit$$

ERbit depends on the buffer structure and technology to estimate how many bit-flips occur to write, to read and to preserve the information. *ELbit* is directly proportional to the tile dimension. For regular 2D-mesh NoCs with square tiles, it is reasonable to consider that *ELHbit* and *ELVbit* have the same value. Therefore, the next equation uses *ELbit* as a simplified representation of *ELHbit* and *ELVbit*. Equation (2) computes the dynamic energy consumed by a single bit traversing a NoC, from tile τ_i to tile τ_j, where η corresponds to the number of routers through where this bit passes.

$$(2) \quad EBit_{ij} = \eta \times ERbit + (\eta - 1) \times ELbit + 2 \times ECbit$$

Let τ_i and τ_j be the tiles to which p_a and p_b are respectively mapped. Then, the dynamic energy consumed by a $p_a \rightarrow p_b$ communication is given by $EBit_{ab} = w_{ab} \times EBit_{ij}$. Equation (3) gives the total amount of NoC's dynamic energy consumption (*ENoC*) that is computed for all bits of all communications between processors ($|W|$).

$$(3) \quad ENoC = \sum_{i=1}^{|W|} EBit_{ab}(i), \ \forall p_a, p_b \in P$$

2.4 Energy Parameters Extraction and Model Validation

To acquire *ERbit*, *ELbit* and *ECbit* values, an initial estimation was performed according to the characterization of Hermes NoC [15] (2D mesh) on a 70nm CMOS technology, which is the target communication architecture used here. Next, a 2×3 NoC described in electrical level was simulated several times, having synthetic patterns as inputs from the local links, simulating hypothetical applications. The same input patterns were applied to the high-level tool that uses Equation (3) to energy consumption estimation. Then the initial values of *ERbit*, *ELbit* and *ECbit* were refined to minimize the average difference between high-level estimation and the electrical level, which is the reference used here. This process permits to achieve high-level estimation of energy consumption with less than 7% of average deviation.

3. METHODOLOGY DESCRIPTION

Figure 4 illustrates the flow used to evaluate partitioning and mapping activities, which is implemented inside CAFES [16], a framework for MPSoC design.

Task partitioning into processors has as entries: (i) *processors list*, which has the name and number of all processors enabling to compute the number of task groups; (ii) *application description*, which has all tasks and their communications; (iii) *CPU occupation* that is a constraint to limit the number of task grouped into the same processor; and (iv) *NoC energy parameters* that is used to compute the energy consumption of a given partition.

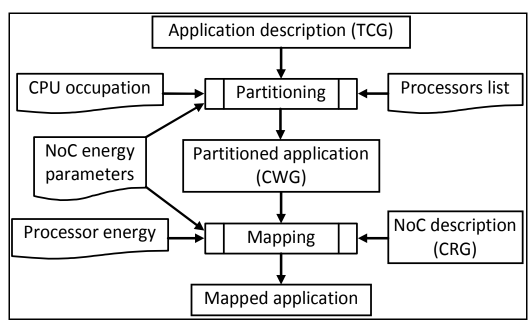


Figure 4 – Partial design flow (partitioning and mapping).

Since partitioning is a NP-complete problem, this work applies a stochastic approach with simulated annealing algorithm (SA) [17]. This one implements a double nested loop. The outer loop tries to find very different partitions aiming to look for global minima. On the other hand, the inner loop explores small partition changes, aiming to find local minima. The algorithm looks for the minimum partitioning cost, which results from the best-searched partition.

The partitioning cost function takes into account the minimization of the overall communication volume. The algorithm tries to achieve a minimum cost, which implies to cluster into the same processor high communicating tasks. In addition, the algorithm tries to balance the CPU occupation through fair distribution of tasks over the available processors. In other words, tasks that communicate most are grouped as far as they do not compromise more than 100% of CPU use for each processor. The CPU occupation constraint is neglected only in cases where there are no other available processors, i.e. the task association to every processor always implies more than 100% of CPU occupation. The partitioning tool generates a CWG description (Section 2.2.1) that contains all processor-tasks associations.

The mapping process also applies the simulated annealing algorithm. The mapping cost function takes into account the communication volume between processors and the NoC energy parameters to compute the energy consumed on a given mapping. Considering a given pair of communicating processors, together with CRG and NoC parameters, the energy consumption is computed through the energy model described on Section 2.2.3. The energy consumption achieved by task running on processor is used only to compute the total energy consumption, but does not affect the mapping choice.

The mapping generates a file containing all processor-tile associations that implies a minimum energy consumption of all evaluated maps.

Partitioning and mapping cost functions use the same NoC energy parameters stated by Equation (2). However, while mapping specifies the exact processor place into the NoC, the partitioning only explores the communication needs, but the number of hops there is between two communicating processors is unknown. In these sense, partitioning cost function uses the concept of *average of hops* that enables to compute the average energy consumption of all possible paths. Let **X** and **Y** be the number of tiles in horizontal and vertical dimension of a NoC, respectively, than Equation (4) computes the total number of hops of all paths that all processors have regarding to XY routing algorithm.

The *average of hops* is computed dividing the summation of all hops of all paths of all processors by the total number of communications, which is stated by Equations (5)(6) and (7).

$$totalHops = \sum_{x=0}^{X-1}\sum_{y=0}^{Y-1}\sum_{i=0}^{X-1}\sum_{j=0}^{Y-1}(|x-i| + |y-j|) \tag{4}$$

$$\#Processors = X \times Y \tag{5}$$

$$maxComm = \#Processors \times (\#Processors - 1) \tag{6}$$

$$hopsAverage = \frac{totalHops}{maxComm} \tag{7}$$

The $hopsAverage$ value is used on Equation (2) in place of η, resulting an average value of $EBit_{ij}$. This one multiplied by the communication volume is the energy consumption estimation of each communication, which is used during the partitioning.

4. METHODOLOGY EXEMPLIFICATION

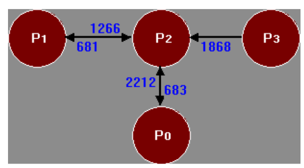
This Section exemplifies the methodology used here. Since this work uses a set of XML tags to capture the parallel application, Figure 5 describes a partial XML structure containing the description of a synthetic parallel application composed by 6 tasks (**T_0**, **T_1**, **T_2**, **T_3**, **T_4** and **T_5**) that are associated to 4 processors (**P_0**, **P_1**, **P_2** and **P_3**), all of the same type (**PowerPC**).

Having as entry the XML description of Figure 5 the partitioning tool generates the following tasks-processor association: {(**P_0**, **T_3**), (**P_1**, **T_5**), (**P_2**, (**T_1**, **T_2**, **T_4**)), (**P_3**, **T_0**)}.

Figure 6 shows a graphical description of the CWG, which is the output description of the partitioning tool. CWG vertices and edges are P = {**P_0**, **P_1**, **P_2**, **P_3**} and W = {(**P_0**, **P_2**), (**P_2**, **P_0**), (**P_1**, **P_2**), (**P_2**, **P_1**), (**P_3**, **P_2**)}, respectively. The edge labels w_{P0_P2} = 2212, w_{P2_P0} = 683, w_{P1_P2} = 1266, w_{P2_P1} = 681 and w_{P3_P2} = 1868 can be easily extracted from Figure 5 with the task-processor associations described above.

```
<PROCESSOR_TYPE type="PowerPC"> <LIST> P0 P1 P2 P3 </LIST></PROCESSOR_TYPE>
<PROCESSOR_TASK_TABLE>
        <TASK id="T0" power="20.68" cpuOccupation="63.82"/>
        <TASK id="T1" power="21.53" cpuOccupation="33.45"/>
        <TASK id="T2" power="36.18" cpuOccupation="27.13"/>
        <TASK id="T3" power="8.75" cpuOccupation="69.18"/>
        <TASK id="T4" power="22.59" cpuOccupation="25.47"/>
        <TASK id="T5" power="16.67" cpuOccupation="55.96"/>
</PROCESSOR_TASK_TABLE>
<TASK_TABLE>    # Here is described the TCG graph
        <SOURCE_TASK source="T0">
                <COMMUNICATION target="T2" volume="1868"/></SOURCE_TASK>
        <SOURCE_TASK source="T1">
                <COMMUNICATION target="T5" volume="681"/>
                <COMMUNICATION target="T2" volume="2183"/></SOURCE_TASK>
        <SOURCE_TASK source="T2">
                <COMMUNICATION target="T1" volume="1516"/></SOURCE_TASK>
        <SOURCE_TASK source="T3">
                <COMMUNICATION target="T1" volume="2212"/></SOURCE_TASK>
        <SOURCE_TASK source="T4">
                <COMMUNICATION target="T3" volume="683"/>
                <COMMUNICATION target="T2" volume="1774"/></SOURCE_TASK>
        <SOURCE_TASK source="T5">
                <COMMUNICATION target="T4" volume="1266"/></SOURCE_TASK>
</TASK_TABLE>
```

Figure 5 – Example of a synthetic application description.



(a)

Processor	Tasks
P0	T3
P1	T5
P2	T1, T2, T4
P3	T0

(b)

Figure 6 – (a) Graphical CWG description of a synthetic application partitioned into four processors; (b) Processors-tasks association.

The mapping has as input the CWG, the topology (CRG) and the energy parameters of the MPSoC. Figure 7 depicts the associations ((τ_1, **P_3**), (τ_2, **P_0**), (τ_3, **P_1**), (τ_4, **P_2**)) generated by mapping.

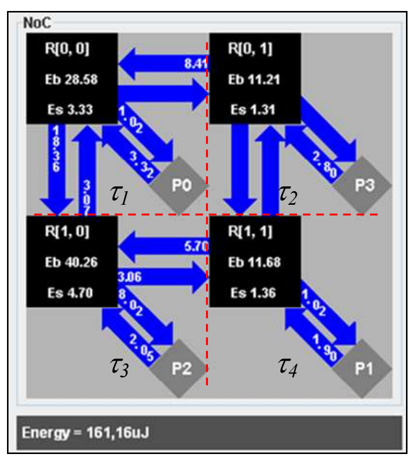


Figure 7 – A processor mapping onto a 2D-mesh NoC architecture with energy consumption annotated inside links and routers. E.g.: inside the router R[0,0], the dynamic energy consumed by the buffers (*Eb*) and switches (*Es*) are 28.58uJ and

3.33uJ, respectively; The dynamic energy consumed in link between router and processor P0 are 1.02uJ and 3.32uJ; The dynamic energy consumed by vertical and horizontal links are 8.41uJ, 3.87uJ and 18.36uJ.

Each link of the communication architecture is associated to an estimated value of the dynamic energy consumption, which depends on the energy parameters and on the quantity of flits, which passes through the communication links. The energy parameters of local links and links between routers are *ECbit* and *ELbit*, respectively. Furthermore, each router contains the energy consumed in buffers (*Eb*) and switches (*Es*). The sum of *Eb* and *Es* is the *ERbit* parameter of Section 2.2.3. Besides, **Energy** provides an overall estimation of the MPSoC energy consumption (161.16 μJ). The energy consumption values are achieved by the characterization of Hermes NoC [15] on a 70nm CMOS technology.

5. EXPERIMENTAL RESULTS

To achieve fair and meaningful results, it is necessary to study a wide range of applications, which is a very time consuming task. Besides, it is hard to find a set of applications, which covers several parallel aspects needed to evaluate MPSoC designs. With this purpose, it was developed a generator of synthetic parallel applications that allows characterizing several application classes according to the experiments.

All results evaluate the influence of partitioning and/or mapping on the energy consumption saving. The results are evaluated in percentage, illustrating how much the partitioning and/or mapping may accomplish the requirements.

Figure 8 and Figure 9 summarize the first set of evaluations, which explores synthetic applications with 50 tasks, fixed power consumption, CPU usage varying randomly from 10% to 80%, 16 processors, and a range of *number of communications* (10%, 20%, 40%, 60%, 80% and 100%) and of *communication volume* (1, 10, 100, 1000 and 10000). The *number of communications* is expressed in percentages - 100% means that all tasks communicate with all other tasks, 0% is the absence of communications, and intermediate values are linearly computed. The combinations of *number of communications* and *communication volume* totalize thirty synthetic applications.

Figure 8 illustrates the effect of *communication volume* variation on energy consumption saving for five *communication volumes* (1, 10, 100, 1000 and 10000). Each dot of the curves contains the average of all energy consumption values computed with experiments varying all six *number of communications* (10%, 20%, 40%, 60%, 80% and 100%) for a given *communication volume*.

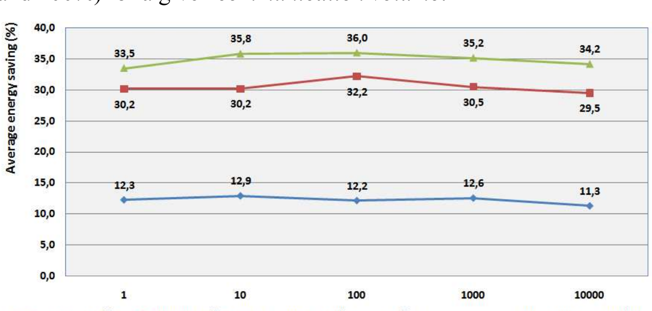


Figure 8 – Partitioning and mapping influence on energy consumption saving for a range of *communication volumes*.

It is a fact that the increase of communication volume raises the energy consumption of the target architecture. However, Figure 8 shows that the average energy saving is similar for all *communication volumes*, independently of design activity. Besides, the mapping is around of three times more efficient, when compared to the partitioning, and the joint effect of both activities is not meaningful, if compared to only mapping results.

Figure 9 illustrates the effect of *number of communications* variation on energy consumption saving. In opposition of Figure 8, each dot concentrates the values of all five-*communication volumes* (1, 10, 100, 1000 and 10000) for a given *number of communications*.

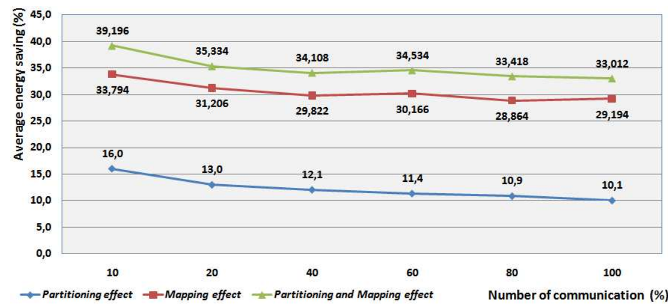


Figure 9 – Partitioning and mapping influence on energy consumption saving for a range of *number of communications*.

The results achieved with *number of communications* variation are similar to the ones achieved with the *communication volume* variation. Nevertheless, the design tasks are more efficient for few communications because there are more scenarios that allow approximating communicating tasks and processors.

Figure 10 and Figure 11 show results of the second set of experiments. This set evaluate synthetic applications with 40% of number of communications, each one having 100 phits (communication volume), fixed power consumption, CPU usage varying randomly from 10% to 80%, and a range of six *NoC sizes* (2x3, 3x3, 3x4, 4x4, 4x5 and 5x5) and of *six number of tasks* (10, 20, 30, 40, 50 and 100). The combinations of *NoC sizes* and *number of tasks* totalize thirty-six synthetic applications.

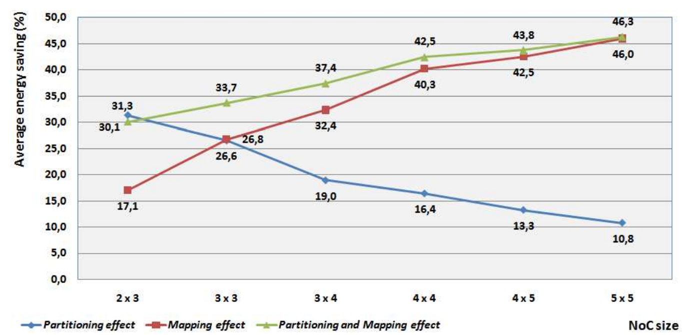


Figure 10 – Partitioning and mapping influence on energy consumption saving for a range of *NoC sizes*.

Figure 10 illustrates the effect of *NoC size* variation on energy consumption saving for six *NoC sizes*: 2x3, 3x3, 3x4, 4x4, 4x5 and 5x5. All NoCs are fully populated, which implies six quantities of processors: 6, 9, 12, 16, 20 and 25, for each NoC size, respectively. Each dot of the curves contains the average of all energy consumption values computed with experiments varying all six *numbers of tasks* (10, 20, 30, 40, 50 and 100) for a given *NoC size*.

Figure 10 shows the importance of partitioning when a meaningful number of tasks are clustered into a unique processor (e.g. 100 tasks grouped into 2x3 processors, implying 16 tasks/processor). On the other hand, when the relation *number of tasks* versus *NoC size* decreases, the partitioning efficiency also reduces and the mapping effectiveness becomes evident.

Figure 11 illustrates the effect of *number of tasks* variation on energy consumption saving. In opposition of Figure 10, each dot concentrates the values of all *NoC size* for a given *number of tasks*.

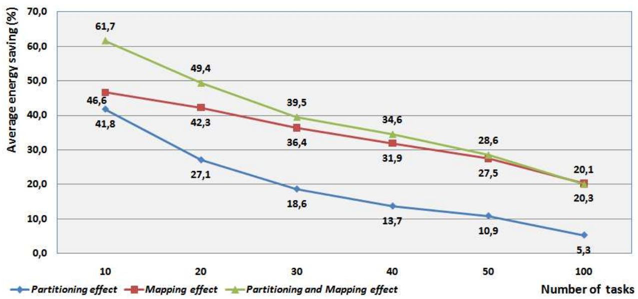


Figure 11 – Partitioning and mapping influence on energy consumption saving for a range of *number of tasks*.

The *number of tasks* increasing reduces the efficiency of mapping and partitioning, which is justified by the reduction of possible partitions - partition algorithm applies load-balancing technique avoiding more than 100% of CPU occupation. On the other hand, a reduced *number of tasks* implies more partitioning and mapping possibilities increasing their influence on the average energy saving.

Last experiment evaluates the partitioning and mapping influence on four real applications: (i) a digital PBX (Privative Branch Exchange); (ii) an image recognition system (IRS); (iii) a distributed algorithm for Romberg integral calculus; and (iv) a multimedia system (MMS). Table 2 depicts some relevant features of these applications.

Table 2 - Characteristics of four real applications.

	PBX	IRS	Romberg	MMS
NoC size (lines x columns)	2 x 3	2 x 3	3 x 4	4 x 4
Number of processors	5	6	10	16
Number of tasks	24	12	30	34
Number of communications	142	53	60	182
Average communication quantity (bytes)	2,334	30,827	35	22,135

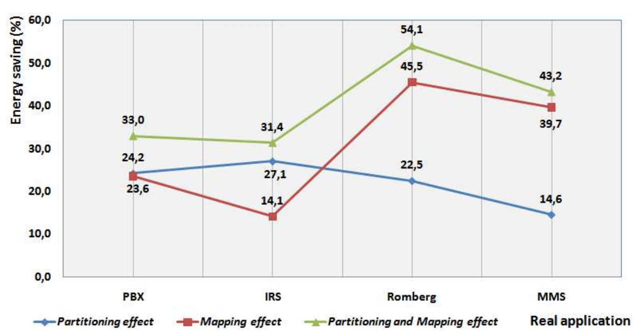


Figure 12 – Partitioning and mapping influence on energy consumption saving for four real applications.

Figure 12 illustrates the partitioning and mapping effect on energy saving for the four real applications characterized in Table 2. The results achieved when considering synthetic applications are sustained. However, two relevant results are pointed out: (i) MMS and Romberg are typically dataflow applications. The dataflow behavior is not well captured by partitioning algorithms, since flows are not specified, nevertheless the mapping effect on energy saving is evident; and (ii) the task partitioning applied on IRS application provides communication-balanced clusters of tasks. This communication balancing reduces the mapping efficiency, since the communication volume does not have a meaningful variation with processors placement.

6. CONCLUSIONS AND FUTURE WORK

This paper explores the effect on energy consumption of the task partitioning into processors and of the processor mapping onto tiles of a NoC-based MPSoC. For almost all experiments, the mapping activity is more energy saving efficient, when compared to partitioning activity, albeit the partitioning effect cannot be neglected. Besides, the joint effect of both activities saves in average 37% of energy. Experiments with several synthetic and real applications attest these results.

In future, we plan to explore the same partitioning and mapping problems for *heterogeneous* NoC-based MPSoC, regarding to not only different processor types, but also different hierarchies of memory.

REFERENCES

[1] Jalier, C. et al. **Heterogeneous vs homogeneous MPSoC approaches for a Mobile LTE modem**. *DATE*, pp.184-189, Oct. 2010.

[2] Le Beux, S. et al. **Combining mapping and partitioning exploration for NoC-based embedded systems**. *JSA*, v.56(7), pp.223–232, Jul. 2010.

[3] Bononi, L. et al. **NoC Topologies Exploration based on Mapping and Simulation Models.** Digital System Design Architectures, Methods and Tools, 10th Euromicro Conference, pp. 543-546, 29-31 Aug. 2007.

[4] Leupers, R. and Castrillon, J.; **MPSoC programming using the MAPS compiler**. *ASP-DAC*, pp.897-902, 18-21 Jan. 2010.

[5] Nedjah, N.; Silva, M., V., C. and Mourelle, L., M. **Customized computer-aided application mapping on NoC infrastructure using multi-objective optimization**. *JSA* v.57(1), pp. 79-94. Jan. 2011.

[6] Tsai, K.; Lai, F.; Pan, C.; Xiao, D.; Tan, H. and Lee, H. **Design of low latency on-chip communication based on hybrid NoC architecture**. NEWCAS Conference, pp.257-260, Jun. 2010.

[7] Youness, H. et al. **A high performance algorithm for scheduling and hardware-software partitioning on MPSoCs**, International Conference on Design & Technology of Integrated Systems in Nanoscale Era. pp. 71-76, Apr. 2009.

[8] Go□hringer, D.; Hu□bner, M.; Benz, M. and Becker, J. **A Design Methodology for Application Partitioning and Architecture Development of Reconfigurable Multiprocessor Systems-on-Chip**. *IEEE Annual International Symposium on Field-Programmable Custom Computing Machines*, pp.259-262, May 2010.

[9] Liu, T.; Zhao, Y.; Li, M. and Xue, C. J.; **Task Assignment with Cache Partitioning and Locking for WCET Minimization on MPSoC**. *International Conference on Parallel Processing*, pp. 573-582, 2010.

[10] Sherwani, N. A.; **Algorithms for VLSI Physical Design Automation**, 2nd. Edition. *Kluwer Academic Publisher*, USA, 1999.

[11] Zwillinger, D.; **Standard Mathematical Tables and Formulae**, 30th Edition. *CRC Press Inc*, USA, 1996.

[12] Hu, J. and Marculescu, R. **Energy-aware mapping for tile-based NoC architectures under performance constraints**. *ASP-DAC*, pp.233-239, Jan. 2003.

[13] Murali, S. and De Micheli, G. **Bandwidth-constrained mapping of cores onto NoC architectures**. *DATE*, pp. 896-901, Feb. 2004.

[14] Ye, T.; Benini, L. and De Micheli, G. **Analysis of power consumption on switch fabrics in network routers**. *DAC*, pp. 524-529, Jun. 2002.

[15] Moraes, F et al. **HERMES: an infrastructure for low area overhead packet-switching networks on chip**. *Integration, the VLSI Journal*, v.38(1), pp. 69-93, Oct. 2004.

[16] Marcon, C. et al. **CAFES: A framework for intrachip application modeling and communication architecture design.** *Journal of Parallel and Distributed Computing*, v.71(5), pp. 714-728, 2011.

[17] Kirkpatrick, S.; Gelatt, C. D. and Vecchi, M. P. **Optimization by simulated annealing**, Science, pp. 671-680, 1983.