Optimal IP Assignment for Efficient NoC-based System Implementation using NSGA-II and MicroGA

Marcus Vinícius Carvalho da Silva¹, Nadia Nedjah¹ and Luiza de Macedo Mourelle²

¹Department of Electronics Engineering and Telecommunications, Engineering Faculty, State University of Rio de Janeiro, Brazil
E-mail: marcus@uerj.br

²Department of System Engineering and Computation, Engineering Faculty, State University of Rio de Janeiro, Brazil

Received: 01/10/08
Revised: 19/05/09

Abstract

Network-on-chip (NoC) are considered the next generation of communication infrastructure, which will be omnipresent in most of industry, office and personal electronic systems. In platform-based methodology, an application is implemented by a set of collaborating intellectual properties (IPs) blocks. In this paper, we use two multi-objective evolutionary algorithms to address the problem of selecting the most adequate set of IPs (from an available library) that best implements the application. The IP selection optimization is driven by the minimization of hardware area, total execution time and power consumption.

Keywords: Network-on-Chip, IP assignment, IP mapping, multi-objective optimization.

1. Introduction

As the integration rate of semiconductors increases, more complex cores for system-on-chip (SoC) are launched. A simple SoC is formed by homogeneous or heterogeneous independent components while a complex SoC is formed by interconnected heterogeneous components. The interconnection and communication of these components form a network-on-chip (NoC). A NoC is similar to a general network but with limited resources, area and power. Each component of a NoC is designed as an intellectual property (IP) block. An IP block can be a general or special purpose such as a processor, memory and DSPs.

Normally, a NoC is designed to run a specific application. This application, usually, consists of a limited number of tasks that are implemented by a set of IP blocks. Different applications may have a similar, or even the same, set of tasks. An IP block can implement more than a single task of the application. For instance, a processor IP block can execute many tasks as a general processor does but a multiplier IP block for floating point numbers can only multiply floating point numbers. The number of IP blocks designers, as well as the number of available IP blocks, is growing up fast.

In order to yield an efficient NoC-based design for a given application, it is necessary to choose the adequate minimal set of IP blocks. With the increase
of IP blocks available, this task is becoming harder and harder. Besides IP blocks careful assignment, it is also necessary to map these blocks onto the NoC available infra-structure, which consists of a set of cores communicating through switches. Of course, a bad mapping can degrade the NoC performance. Different optimization criteria can be pursued depending on how much information details is available about the application and IP blocks.

Usually, the application is viewed as a graph of tasks called task graph (TG). The features of IP blocks can be obtained from their companion documentation. The task assignment and IP block mapping key research problems for efficient NoC-based designs. These two problems can be solved using multi-objective optimizations in which some of the objectives are conflicting. Because of their nature, both IP assignment and IP mapping are classified as NP-hard problems. Normally, deterministic techniques are not viable to solve such problems so we used multi-objective evolutionary algorithms (MOEAs) with specific operators and objective functions. For this purpose, one needs to select the best minimal set of objectives to be optimized. The wrong set of optimized objectives can lead to average instead of best results. In this work, we select to optimize three non-collaborative objectives, which are the hardware required for the implementation of the NoC-based system, the response time imposed and the power consumption by the system while operating. Note that the non-linear relationship between these three objectives.

In this paper, we propose a multi-objective evolutionary-based decision support system to help NoC designers with respect to the IP assignment stage. For this purpose, we propose a structured representation of the TG and an IP repository that will feed data into our tool. We use data from The Embedded Systems Synthesis benchmarks Suite (E3S) as our IP repository. The E3S is a collection of TGs, representing real applications based on embedded processors from the Embedded Microprocessor Benchmark Consortium (EEMBC). It was developed to be used in system-level allocation, assignment, and scheduling research. We use two MOEAs: NSGA-II and microGA. Both of these algorithms were modified according to some prescribed NoC design constraints. Note that as far as the authors are concerned, this is this application is novel and the results obtained are competitive.

The rest of the paper is organized as follows: First, in Section 2, we present the related work available so far. Subsequently, in Section 3, we describe a structured TG and IP repository model based on the E3S data. After that, in Section 4, we present the IP assignment problem. Then, in Section 6, we sketch the two MOEAs used in this work, individual representations and objective functions for the optimization process. Later, in Section 7, we show some experimental results. Last but not least, in Section 8, we draw some conclusions and outline new directions for future work.

2. Related Work

The problems of allocating IP blocks to application’s tasks and mapping those blocks into a NoC physical space have been addressed in some previous studies with different emphasis. Some of these research works did not take into account the multi-objective nature of these problems and adopted a single objective optimization approach.

Hu and Marulescu proposed a branch and bound algorithm which automatically maps IPs/cores into a mesh based NoC architecture that minimizes the total amount of consumed power by minimizing the total communication among the used cores. Specified constraints through bandwidth reservation were defined to control communication limits.

Lei and Kumar proposed a two step genetic algorithm for mapping the TG into a mesh based NoC architecture that minimizes the execution time. In the first step, they assumed that all communication delays are the same and selected IP blocks based on the computation delay imposed by the IPs only. In the second step, they used real communication delays to find an optimal binding of each task in the TG to specific cores of the NoC.

Murali and De Micheli addressed the problem under the bandwidth constraint with the aim of minimizing communication delay by exploiting the
possibility of splitting traffic among various paths. Splitting the traffic increases the size of the routing component at each node but the authors were not worried about size.

Zhou et al.\cite{16} suggested a multi-objective exploration approach, treating the mapping problem as a two conflicting objective optimization problem that attempts to minimize the average number of hops and achieve a thermal balance. The number of hops is incremented every time a data cross a switch before reaching its target. They used NSGA\textsuperscript{15}, multi-objective evolutionary algorithm. They also formulated a thermal model to avoid hot spots, which are areas with high computing activity.

Jena and Sharma\textsuperscript{a} addressed the problem of topological mapping of IPs/cores into a mesh-based NoC in two systematic steps using the NSGA-II \textsuperscript{3}. The main objective was to obtain a solution that minimizes the energy consumption due to both computational and communicational activities and also minimizes the link bandwidth requirement under some prescribed performance constraints.

As a recent field of research, the available literature about NoC-based design optimization is scarce. The aforementioned works represent the state of the art of this field. In (Ref 8, Ref 10 and Ref 11), only one objective was considered and only\textsuperscript{10} used an evolutionary approach. In (Ref. 9 and Ref. 15), two objectives were considered and both adopted a MOEA to solve the problem.

3. Task Graph and IP Repository Models

In order to formulate the IP assignment problem, it is necessary to introduce a formal definition of an application. An application can be viewed as a set of tasks that can be executed sequentially or in parallel. It can be represented by a directed graph of tasks, called task graph.

**Definition 1.** A Task Graph (TG) \( G = G(T, D) \) is a directed graph where each node represents a computational module in the application referred to as task \( t_i \in T \). Each directed arc \( d_{i,j} \in D \), between tasks \( t_i \) and \( t_j \), characterizes either data or control dependencies.

Each task \( t_i \) is annotated with relevant information, such as a unique identifier and type of processing element (PE) in the network. Each \( d_{i,j} \) is associated with a value \( v(d_{i,j}) \), which represents the volume of bits exchanged in communication between tasks \( t_i \) and \( t_j \).

Once the IP assignment is performed, each task is associated with an IP identifier. The result of this stage is a graph of IPs representing the PEs responsible of executing the application.

**Definition 2.** An Application Characterization Graph (ACG) \( G = G(C,A) \) is a directed graph, where each vertex \( c_i \in C \) represents a selected IP/core and each directed arc \( a_{i,j} \) characterizes the communication process from core \( c_i \) to core \( c_j \).

Each \( a_{i,j} \) can be tagged with IP/application specific information, such as communication rate, communication bandwidth or a weight representing communication cost.

A TG is based on application features only while an ACG is based on application and IP features, providing us with a much more realistic representation of the application in runtime on a NoC structure. In order to be able to bind application and IP features, at least one common feature is required in both of the IP and TG models.

The E3S (0.9) Benchmark Suite\textsuperscript{4} contains the characteristics of 17 embedded processors. These processors are characterized by the measured execution times of 47 different type of tasks, power consumption derived from processor datasheets, and additional information, such as die size, price, clock frequency and power consumption during idle state. In addition, E3S contains task graphs of common tasks in auto-industry, networking, telecommunication and office automation. Each one of the nodes of these task graphs is associated with a task type. A task type is a processor instruction or a set of instructions, e.g., FFT, inverse FFT, floating point operation, OSPF/Dijkstra\textsuperscript{6}, etc. If a given processor is able to execute a given type of instruction, so that processor is a candidate to receive a resource in the NoC structure and would be responsible for the execution of one or more tasks.

Published by Atlantis Press
Copyright: the authors
3.1. XML Representation

The E3S Benchmark Suite contains rich data about embedded processors and some common applications. TGFF, a random TG generator based on E3S processors, generates TGs with parallel and sequential tasks, nodes with IP types and other important features. Both, E3S and TGFF, are text files. We use XML Schema to model the TG and IP repository. At this point, no standard schema for NoC design is available, so the XML structure for both representations reflects the features available from E3S processors and applications.

3.2. Task Graph Representation

Here, we represent TGs using the XML. A TG is divided in three major elements: taskgraph, nodes and edges. The taskgraph element is the TG itself which contains nodes and edges. The nodes element includes a node element for each task of the TG and the edges element includes an edge element for each arc in the TG. Each node has two main attributes: an unique identifier (id) and a task type (type), chosen among the 47 different types of tasks present in the E3S. Each edge has four main attributes: an unique identifier (id), the id of its source node (src), the id of its target node (tgt) and an attribute representing the communication cost imposed (cost). Fig. 1 shows a simple TG and Fig. 2 its corresponding XML representation.

![Fig. 1. Example of task graph](image)

3.3. IP Repository Representation

The IP repository is divided into two major elements: the repository and the ips elements. The repository is the IP repository itself. Recall that the repository contains different non-general purpose embedded processors and each processor implements up to 47 different types of operations. Not all 47 different types of operations are available in all processors. Each type of operation available in each processor is represented by an ip element. Each ip is identified by its attribute id, which is unique, and by other attributes such as taskType, taskName, taskPower, taskTime, processorID, processorName, processorWidth, processorHeight, processorClock, processorIdlePower and cost. The common element in TG and IP repository representations is the type attribute. Therefore, this element will be used to bind an ip to a node. Fig. 3 shows a simplified XML structure representing the IP repository. The repository contains IPs for digital signal processing, matrix operations, text processing and image manipulation.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<taskgraph>
  <nodes>
    <node id="0" type="45" />
    <node id="1" type="21" />
    <node id="2" type="28" />
    <node id="3" type="32" />
  </nodes>
  <edges>
    <edge id="0" src="0" tgt="1" cost="5" />
    <edge id="1" src="1" tgt="3" cost="4" />
    <edge id="2" src="1" tgt="2" cost="3" />
    <edge id="3" src="2" tgt="3" cost="2" />
  </edges>
</taskgraph>
```

![Fig. 2. TG XML structure](image)

```xml
<?xml version="1.0" encoding="UTF-8"?>
<repository>
  <ips>
    <ip id="10" type="0" procID="3" />
    <ip id="23" type="38" procID="5" />
    <ip id="68" type="12" procID="14" />
    <ip id="99" type="47" procID="17" />
  </ips>
</repository>
```

![Fig. 3. IP repository XML structure](image)

These simplified and well-structured representations are easily intelligible, improve information processing and can be universally shared among different NoC design tools.
4. The IP Assignment Problem

The platform-based design methodology for SoC is based on the components reuse philosophy to increase reusability and to reduce the time-to-market of new designs. The designer of NoC-based systems faces two main problems: selecting the right set of IPs that optimize the execution of a given application and finding the best physical mapping of these IPs into the NoC structure.

The main objective in this paper is to select a set of IPs, from the IP repository, that minimizes the NoC consumption of power, area occupied and execution time. Different IP characteristics have to be analyzed to optimize each one of these objectives. For example, to optimize the execution time, the processor clock and the execution time of each task have to be considered simultaneously. At this step, no information about physical allocation of IPs is available so optimization must be done based on TG and IP information only. The result of this step is the set of IPs that should maximize the NoC performance, without consideration of their respective physical allocation in the NoC internal structure. The result of this optimization produces an ACG from the TG, where each node has an IP associated with it.

5. The choice of optimization Objectives

Different objectives may be considered in the IP assignment problem. If the improvement of an objective leads to a deterioration of another one (e.g. maximizing clock frequency increases power consumption), the objectives are said to be concurrent. On the other hand, if the improvement of an objective leads to also an improvement of another one, the objectives are said to be collaborative. Optimization problems with concurrent and collaborative objectives are called Multi-objective Optimization Problems (MOPs). In such problems, all collaborative objectives should be grouped and one single member of the group should be selected to be optimized, aiming at the optimization of the whole group. However, concurrent objectives should all be considered during the optimization process. The best solution for a MOP is the solution with the adequate trade-off between all concurrent objectives.

Table 1 helps choosing the minimal set of objectives to be considered in IP assignment optimization stage. A up/down arrow in entry for objectives $i \times j$ means that an increase/reduction with respect to objective $i$ also leads to and increase/reduction with respect to objective $j$.

Table 1. Concurrent and Collaborative Objectives

<table>
<thead>
<tr>
<th>Area</th>
<th>Cost</th>
<th>Clock</th>
<th>Time</th>
<th>Power</th>
<th>#PEs</th>
</tr>
</thead>
<tbody>
<tr>
<td>↓</td>
<td>↓</td>
<td>↓</td>
<td>↓</td>
<td>↓</td>
<td>↓</td>
</tr>
<tr>
<td>Cost</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Clock</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Time</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Power</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>#PEs</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

For instance, the last column of Table 1, which characterizes objective #PE (i.e. number of processor elements), indicates that a reduction with respect to this objective yields a reduction with respect to both area and cost. Therefore, those three objectives are considered collaborative. This is the same case for objective time and clock frequency. However, in the penultimate column of Table 1, which characterizes objective power, indicates that a reduction in power leads to an increase in both time and clock frequency. Note that objective power must be minimized. Therefore, objective power is considered concurrent with both objective time and clock frequency. As a conclusion, the adequate trade-off can be achieved using only minimization functions of objectives area, execution time, power consumption.

6. Multi-objective evolutionary approach

The search space for a “good” IP assignment for a given application is defined by the possible combinations of IPs available in the repository. Within a repository of $N$ IPs, we have a domain size of $N!$. Among the huge number of solutions, it is possible to find many solutions equally good. Deterministic approaches do not deal very well with MOPs in huge non-continuous search space. The domination concept introduced by Pareto is necessary to classify solutions. In order to deal with such a big search space and the trade-off between solutions in a reasonable time, a multi-objective evolutionary approach is adopted.

The core of the proposed tool offers the utilization of two well-known and well-tested MOEAs: NSGA-II and microGA. Both adopt the domination concept with a ranking schema for classification. The ranking process separates solutions in Pareto fronts where each front corresponds to a given rank. Solutions from rank one, which is the Pareto-optimal front) are equally good and better than any other solution from Pareto fronts of higher ranks.

NSGA-II features a fast and elitist ranking process
that minimizes computational complexity and provides a good spread of solutions. The elitist process consists in joining parents and offspring populations and diversity is achieved using the crowded-comparison operator\(^3\). MicroGA works with a very small population (3 to 5 individuals), which makes it very fast. A bigger population is stored on a population memory divided in replaceable and non-replaceable areas. The elitist process consists of storing the best solutions on an external memory and diversity is achieved using an adaptive grid\(^4\).

The basic workflow of both algorithms is almost the same. They start with a random population of individuals, where each individual represents a solution. Each individual is associated with a rank. The selection operator is applied to select the parents. The parents pass through crossover and mutation operators to generate an offspring. A new population is created and the process is repeated until the stop criterion is reached.

Once the IP assignment is complete, we have a set of individuals (or a single one) with rank one and those are the best individuals for the available information in the IP repository.

### 6.1. Representation and Genetic Operators

The individual representation is shown in Fig. 4. The selection, crossover and mutation operators were modified based on the developed individual chromosome representation. Tournament selection, one-point crossover and simple mutation were used. The chromosome is formed by a set of genes and each one represents a node id from the TG. Initially, a random IP id is assigned to each node, with the constraint of the IP type. The crossover operator, without any constraint, can only produce feasible individuals because the order of genes is not changed. The mutation is controlled by IP type constraint to avoid selecting a random IP, from IP repository, of different type.

![Gene Encoding](image)

**Fig. 4. Chromosome encoding for IP assignment step**

### 6.2. Objective Function

During the evolutionary process, the fitness of each one of the previously selected objectives (i.e. area, time, and power) must be efficiently computed for each solution.

#### 6.2.1. Area

In order to compute the area required, it is necessary to know the area of the selected processors. As a processor can be responsible for more than one task, each ACG node must be visited in order to check the number of processor elements. Grouping the nodes of same processorID attribute allows us to implement this verification. The area is computed adding up all the processor’s areas. Note that this is not the sum of IP’s areas. Equation 1 shows how to compute the required area to use a given IP assignment, wherein function Proc( ) provides the set of processors used in a given ACG and area\(_{pe}\) is the required area for processor \(pe\) in

\[
\text{Area} = \sum_{\text{pe} \in \text{Proc}(\text{ACG})} \text{area}_{\text{pe}} \tag{1}
\]

#### 6.2.2. Execution Time

In order to compute the execution time imposed, it is necessary to find the critical path of the ACG. The critical path can be found visiting all nodes of all paths and recording the execution time of the slowest path. Equation 2 is computed using a recursive function that implements a depth-first search, wherein function Paths( ) provides all possible paths of a given ACG and time\(_t\) is the required time for task \(t\) in

\[
\text{Time} = \max_{\text{p} \in \text{Paths}(\text{ACG})} \left( \sum_{\text{te} \in \text{p}} \text{time}_{\text{t}} \right) \tag{2}
\]

#### 6.2.3. Power Consumption

In order to compute the power consumption, the task power of all nodes of the ACG is summed up. In Equation 3, \(power_t\) is the required power consumption to execute task \(t\) on the specified processor.

\[
\text{Power} = \sum_{\text{te} \in \text{ACG}} \text{power}_{\text{t}} \tag{3}
\]

### 7. Results

First of all, the implementation of both algorithms (i.e. NSGA-II and microGA) was used to solve mathematical MOPs and the results were compared with the expected results. Fig. 5 shows results of both algorithms for
a two-objective optimization problem (called KUR) proposed by Kursawe and used by Deb and Coello to validate NSGA-II\(^3\) and microGA\(^1\), respectively. Both algorithms converged to the true Pareto-front. As expected, NSGA-II found a higher diversity of solutions while microGA converged much faster. The parameters used for these tests were the same as those originally used by the authors\(^3,1\).

For NoC optimization, only the individual representation and the objective functions were altered, keeping the ranking, selection, crossover and mutation operators unchanged. Different TGs generated with TGFF\(^5\) and from E3S were used. The TGs included sequential and parallel tasks.

Many simulations were used to find out the right values to set up the parameters of NSGA-II and microGA. For the former, we used a population size of 600, mutation probability of 0.01, crossover probability of 0.8, tournament size of 50 and run the algorithm for 50 generations. For the latter, we used a population memory size of 1000 with 70% for the replaceable fraction, micro population of 4 individuals, mutation probability of 0.02, crossover probability of 0.09, tournament size of 2, external memory of 200, nominal convergence of 4, replacement cycle of 100, bisection of 5, and run the algorithm for 3000 generations.

The application, represented as a TG in Fig. 6, was generated by the TGFF\(^5\). This TG is interesting because of the five levels of parallelism, formed by the mirrored left-right side nodes.

The evolutionary process performed for the IP assignment of the TG of Fig. 6 was able to discover 25 distinct optimal IP assignments. The Pareto-optimal solutions together with the respective fitness with respect to each of the considered objectives of the solutions are listed in Table 2. Fig. 7 represents the \textit{time × area} trade-off, Fig. 8 depicts the \textit{power × time} trade-off and Fig. 9 plots the \textit{power × area} trade-off. As we can observe, comparing the dots against the line of interpolation, the trade-off between time and area and between power and time is not so linear as the trade-off between power and area. Fig. 7 shows that solutions that occupies more area tend to spend less time of execution because of the better distribution of parallel tasks. Fig. 8 shows that solutions that spend less time of execution tend to consume more power because of IP’s features (i.e. higher clock) and physical effects like more inner electrons activity. Fig. 9 shows a linear relation between consumption power and area occupied. Those values and units are based on E3S Benchmark Suite\(^4\).
Fig. 7. Trade-off time \times area for TG in Fig. 6

Fig. 8. Trade-off power \times time for TG in Fig. 6

Fig. 9. Trade-off power \times area for TG in Fig. 6

8. Conclusions

The problem of allocating IPs into an application represented as a task graph is an NP-hard key research problems in NoC design.

Table 2. Optimal IP assignment for the task graph of Fig. 6

<table>
<thead>
<tr>
<th>solution</th>
<th>set of IPs</th>
<th>time (s)</th>
<th>area (m²)</th>
<th>power (W)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1462,722,458,378,490,523,240,0,855,864,379,637,110,721,661,239]</td>
<td>0.1233</td>
<td>14.8935</td>
<td>62.9500</td>
</tr>
<tr>
<td>6</td>
<td>1462,937,458,378,860,523,240,0,855,401,379,215,399,721,661,719]</td>
<td>0.1290</td>
<td>14.6892</td>
<td>36.4250</td>
</tr>
<tr>
<td>7</td>
<td>1462,937,458,378,490,523,240,0,855,864,691,13,382,721,661,239]</td>
<td>0.1292</td>
<td>10.8422</td>
<td>34.6250</td>
</tr>
<tr>
<td>8</td>
<td>1462,937,458,378,490,523,240,0,855,592,379,215,399,721,620,719]</td>
<td>0.1297</td>
<td>9.9991</td>
<td>46.5250</td>
</tr>
<tr>
<td>9</td>
<td>1462,937,458,378,490,523,240,0,855,401,379,215,399,721,620,719]</td>
<td>0.1298</td>
<td>9.9982</td>
<td>35.0520</td>
</tr>
<tr>
<td>10</td>
<td>[369,937,458,378,860,523,240,0,855,401,379,215,399,721,620,719]</td>
<td>0.1298</td>
<td>10.0913</td>
<td>34.5200</td>
</tr>
<tr>
<td>11</td>
<td>[866,92,938,378,490,523,240,0,855,401,379,215,399,721,94,719]</td>
<td>0.1319</td>
<td>9.8334</td>
<td>41.8750</td>
</tr>
<tr>
<td>12</td>
<td>1462,92,938,378,490,523,240,0,855,401,379,215,399,721,94,719]</td>
<td>0.1319</td>
<td>9.9856</td>
<td>40.9500</td>
</tr>
<tr>
<td>14</td>
<td>[369,138,938,378,490,523,240,0,855,864,691,13,382,936,724,239]</td>
<td>0.1486</td>
<td>8.0183</td>
<td>32.7200</td>
</tr>
<tr>
<td>15</td>
<td>[866,138,938,378,490,523,240,0,855,112,691,13,382,936,724,239]</td>
<td>0.1486</td>
<td>7.9146</td>
<td>46.5500</td>
</tr>
<tr>
<td>16</td>
<td>[864,92,938,378,490,523,240,0,855,66,691,13,382,936,724,239]</td>
<td>0.1502</td>
<td>8.0124</td>
<td>28.7200</td>
</tr>
<tr>
<td>17</td>
<td>[866,92,938,378,490,523,240,0,855,66,691,13,382,936,724,239]</td>
<td>0.1502</td>
<td>7.9112</td>
<td>38.5500</td>
</tr>
<tr>
<td>18</td>
<td>[866,92,938,378,490,523,240,0,855,401,691,13,382,936,724,239]</td>
<td>0.1505</td>
<td>3.8679</td>
<td>18.7000</td>
</tr>
<tr>
<td>19</td>
<td>[866,92,938,378,490,523,240,0,855,401,691,13,382,936,724,239]</td>
<td>0.1505</td>
<td>4.9264</td>
<td>16.1700</td>
</tr>
<tr>
<td>20</td>
<td>[866,92,938,378,490,523,240,0,855,401,489,13,382,936,724,239]</td>
<td>0.1821</td>
<td>5.8873</td>
<td>17.7700</td>
</tr>
<tr>
<td>21</td>
<td>[369,722,458,378,860,847,240,0,855,494,211,13,382,936,243,239]</td>
<td>0.2011</td>
<td>4.0635</td>
<td>17.3600</td>
</tr>
<tr>
<td>22</td>
<td>[369,937,458,378,860,847,450,0,392,864,211,13,382,936,243,239]</td>
<td>0.2075</td>
<td>5.0214</td>
<td>16.3400</td>
</tr>
<tr>
<td>23</td>
<td>[849,937,458,378,490,847,450,0,855,494,691,13,382,936,243,239]</td>
<td>0.2076</td>
<td>3.9611</td>
<td>16.0400</td>
</tr>
<tr>
<td>24</td>
<td>[369,937,458,378,860,847,450,0,855,494,211,13,382,936,243,239]</td>
<td>0.2076</td>
<td>4.0693</td>
<td>15.4400</td>
</tr>
<tr>
<td>25</td>
<td>[369,937,458,378,860,847,450,480,855,864,211,695,382,936,724,239]</td>
<td>0.2215</td>
<td>3.1033</td>
<td>15.8400</td>
</tr>
</tbody>
</table>

1 (x10⁻⁵m²), 2 (x10⁻¹W)
In this paper, we proposed a decision support system based on MOEAs to help NoC designers to select a set of IPs from a repository of IPs. The use of two different MOEAs consolidates the obtained results. Structured and intelligible representations of a NoC, a TG and of a repository of IPs were proposed. These can be easily extended to different NoC applications. Despite of the fact that we have adopted E3S Benchmark Suite as our repository of IPs, any other repository could be used and modeled using XML, making this tool compatible with different repositories.

Future work is four-fold: tackling the IP mapping problem; adopting a dynamic topology strategy to try to evolve the most adequate topology for a given application; exploring the use of different objectives based on different repositories and proposing an interfacing mechanism with a hardware description simulator to integrate our tool to the NoC design platform.

References