

Efficient Application Mapping onto Three-Dimensional Network-on-Chips Using Multi-Objective Particle Swarm Optimization

Maamar Bougherara[1,3], Nadia Nedjah[2(✉)], Djamel Bennouar[3], Rebiha Kemcha[3,4], and Luiza de Macedo Mourelle[5]

[1] LIMPAF Laboratory, Bouira University, Bouira, Algeria
[2] Department of Electronics Engineering and Telecommunications, State University of Rio de Janeiro, Rio de Janeiro, Brazil
nadia@eng.uerj.br
[3] Département d'Informatique, Ecole Normale Supérieure Kouba, Algiers, Algeria
[4] LIMOSE Laboratory, Boumerdes University, Boumerdes, Algeria
[5] Department of Systems Engineering and Computation, State University of Rio de Janeiro, Rio de Janeiro, Brazil

Abstract. Network-on-chip (NoC) is considered the next generation of communication in embedded system. In this case, an application is implemented by a set of collaborative intellectual propriety blocks (IPs). The selection of the most suited block from a library of IPs as well as their physical mapping onto the three-dimensional Network on-chip infrastructure to implement efficiently the application are two NP-complete problems. In this paper, we propose to use Multi-Objective Particle Swarm Optimization (MOPSO) to yield the best selection of IP and there physical mapping of a given application on three-dimensional topologies. In this purpose, MOPSO is exploited to obtain the personalized result for the application at hand. Only the used resources, switches and channels by the application mapping are part of the customized implementation platform. The optimization is driven by the minimization of required hardware area, the imposed execution time and the necessary power consumption of the final implementation.

Keywords: 3D NoCs · IP mapping · Multi-objective optimization · MOPSO

1 Introduction

Systems-on-Chip (SoC) integrate a complete system into a single chip. The number of integrated components continues to increase to meet the requirements of today's applications. Thereby, the design of SoCs focuses in communications more than computations [1]. Network-on-Chip (NoC) emerged as a viable

© Springer Nature Switzerland AG 2019
S. Misra et al. (Eds.): ICCSA 2019, LNCS 11620, pp. 654–670, 2019.
https://doi.org/10.1007/978-3-030-24296-1_53

solution to on-chip communication bottlenecks. It's similar to a general network but with limited resources such as area, bandwidth and power.

A NoC consists of set of *resources* (R) and *switches* (S). Each resource of the NoC is an Intellectual Property (IP) block that can be a general or special purpose such as a processor, memory and DSP [1]. A NoC is designed to run a specific application. This kind of applications is generally limited by the number of the tasks that are implemented in IP block. An IP block can implement many tasks and execute them as a general-purpose processor does. In contrast, an IP block is conditioned to execute some types of tasks. Therefore, it is necessary to choose the adequate IP block before designing an efficient NoC-based system for any application. The choice becomes harder when the number of tasks increases. It is also necessary to map these blocks onto the NoC available infrastructure. The latter consists of a set of cores communicating using switches.

Resources and switches are connected by *links*. The pair (R, S) forms a *tile*. The simplest way to connect the available resources and switches is arranging them using topologies, the two-dimensional mesh topology is the most popular used in NoC due to its simplicity however the two dimensional Noc has not been thought to meet the requirement of System-on-Chip in many aspects. Three-dimensional network on chip (3D NoC) is an effective solution to the problem of interconnection complexity of large-scale SoC. 3D NoC breaks the limitations on performance and size of two-dimensional Network on Chip (2D NoC) by using integrated circuit stacking technology.

A three-dimensional (3D) mesh is implemented by stacking several layers of 2D mesh on top of each other and providing vertical links for interlayer communication. The vertical interconnections, Through-SiliconVia (TSV) [18] can be a potential candidate. A switch is able to buffer and route messages between resources. Each switch is connected to up to six other neighboring switches through input and output channels. While a channel is sending data another channel can buffer incoming data. Figure 1 shows the architecture of a 3D mesh-based NoC, where each resource contains one or more IP blocks. In this figure, RNI stands for Resource Network Interface, D for DSP, M for Memory, C for Cache, P for Processor, FP for Floating-Point unit and Re for a Reconfigurable block.

Different optimization criteria can be pursued depending on the detailed information of the application and IP blocks. The application is viewed as a graph of task called Task Graph (TG). The features of an IP block can be determined from its companion library [2]. The objectives involved in task assignment and IP block mapping are multiple and have to be optimized simultaneously. Some of these objectives are conflicting because of their nature. So, IP assignment and IP mapping are classified as NP-hard problems [3]. Thus, it is mandatory to use Multi-Objective Evolutionary Algorithms (MOEAs) with specific objective functions.

In this paper, we propose a multi-objective evolutionary based decision to help NoC designers to select the most suited IP blocks used during the assignment step and their physical mapping in network on chip in three-dimensional topology.

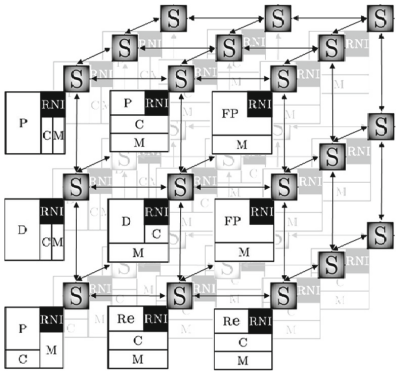


Fig. 1. 3D Mesh-based NoC with 27 resources

For this aim, we use the structure representation of TG in [4] and IP repository data from the Embedded Systems Synthesis benchmarks Suite (E3S) [5] as an IP library for the proposed tool. We use Multi-objective Particle Swarm Optimization (MOPSO) algorithm [6], which was modified to suit the specificities of the assignment and Mapping problems and also to guarantee the NoC designer's constraints.

The rest of the paper is organized as follows: First, in Sect. 2, we present the related work. Subsequently, in Sect. 3, we present the model used for the application structure and IP repository. After that, in Sect. 4 we provide an overview of the assignment and mapping problems. Then, in Sect. 5, we present the for multi-objective approach using MOPSO algorithm used in this work. After that, in Sect. 6, we define the objective functions to be optimized. Later, in Sect. 7, we show some experimental results. Finally, in Sect. 8, we conclude the work and discuss new directions for future work.

2 Related Work

The problems of mapping IP blocks into a 3D mesh-based NoC have been addressed in some previous works. A Branch and Bound algorithm for smart energy aware mapping of tasks have been proposed in [12], the energy model in 2D in [1] was extended for 3D mesh-based NoC topology. In [11] suggest thermal and power task mapping on the platform of 3D mesh-based NoC based on fuzzy logic. A genetic algorithm to build 3D mesh NoC with reduced power consumption from a given application in [17]. The algorithm was tested and validated in an architecture of 32 microprocessors. In [10]. Quantum-behaved particle swarm optimization algorithm is applied to reduce the power consumption, the Simulation results show that this algorithm has faster convergence speed with much better optimization performance compared with particle swarm algorithm. In [9] other varient of particle swarm optimization is used, based on diversity-controlled quantum-behaved particle swarm optimization (DCQPSO).

the results show that this algorithm is able to maintain a stable power optimization efficiency and converges faster. For the first time, a novel meta-heuristic algorithm mimicking hunting behaviors of bats is applied in low power mapping methods in [8]. In [7], the author combines a genetic algorithm with simulated annealing, the Experiments show that ISAGA has a good effect in solving 3D NoC low-power mapping.

All the aforementioned works adopted one single objective. In the following, we describe some works wherein the mapping problem is handled via multi-objective optimization. Thermal and communication aware mapping for 3D NoCs can be performed using genetic algorithms in [13]. A multi-objective rank-based genetic algorithm (RMGA) for 3D NoCs is proposed in [14]. In this paper, two different models have been used for packet latency under no congestion and congestion situations respectively. A multi-objective immune algorithm (MIA) used in [15], Latency and power consumption are adopted as the target multi-objective functions constrained by the heating function. The author in [16] propose a centralized 3D mapping (C3Map) using a new octahedral traversal technology and realized the H-C3Map which is formed from C3Map and attractive and repulsive particle swarm optimization (ARPSO) which reduced energy and latency.

3 Application and IP Repository Models

In order to formulate the IP assignment and mapping problems, it is necessary to define the application internal model that will be used. An application can be represented by a set of tasks that can be executed sequentially or in parallel. It is represented by a directed acyclic graph, called Task Graph (TG) [20].

Definition 1 [19]**:** A task graph $G = (T, D)$ is a directed graph, wherein a node represents a task $t_i \in T$ and a directed arc $d_{ij} \in D$ between tasks t_i and t_j represents the data dependency between these tasks. The arc label $v(d_{ij})$ characterizes the volume of bits exchanged in communication between tasks t_i and t_j.

Definition 2 [21]**:** An application characterization graph $G = G(C, A)$ is a directed acyclic graph, wherein each vertex $c_i \in C$ represents a selected IP block and each directed arc $a_{ij} \in A$ characterizes the communication process between c_i and c_j.

In order to help NoC designers and standardize the proposed solution, we structured the used application repository, which is the E3S benchmark suite [22] using XML. XML schema provide a neat and well-accepted model for the task graph and IP repository.

A TG is divided into three major elements: The task graph element is the TG itself, which contains tasks and edges. A task element includes a task element for each task of the TG while an edge element includes an edge element for each arc in the TG. Each task has two main attributes: an unique identifier (*code*)

and a task type (*type*), chosen among the 46 different types of tasks included in the E3S library [22]. Each edge has four main attributes: an unique identifier (*id*), an identifier of its source node (*src*), another of its target node (*tgt*) and an attribute representing the communication cost imposed (*cost*). Figure 2 shows the XML representation of a simple TG of E3S.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<task_graph>
  <tasks>
    <task code ="0" name="src"     type="45" />
    <task code ="1" name="text"    type="44" />
    <task code ="2" name="sink"    type="45" />
    <task code ="3" name="rotate" type="43"/>
    <task code ="4" name="dith"    type="42" />
  </tasks>
  <EDGES>
    <edge name="a0_0" from="0" to="1" cost="1000"/>
    <edge name="a0_1" from="0" to="3" cost="7070"/>
    <edge name="a0_2" from="3" to="4" cost="7070"/>
    <edge name="a0_3" from="4" to="2" cost="7070"/>
    <edge name="a0_4" from="1" to="2" cost="1000"/>
  </edges>
</task_graph>
```

Fig. 2. XML code used to describe tasks

The IP repository is divided into two major elements: the repository and the IPs elements. The repository is the IP repository itself. It is noteworthy to point out that the repository contains different non-general purpose embedded processors and each processor implements up to 46 different types of operations. Not all 46 different types of operations are available in all processors. Each type of operation available to be run in each processor is represented by an IP element. Each IP is identified by its unique identifier. It also includes other attributes such as taskType, taskName, taskPower, taskTime, processorID, processorName, processorWidth, processorHeight, processorClock, processorIdle, Area, Power and cost. The common element in TG and IP repository representations is the type attribute. Therefore, this element will be used to link an IP to a node. Figure 3 shows a simplified XML structure representing an IP repository. The repository contains IPs for digital signal processing, matrix operations, text processing and image manipulation.

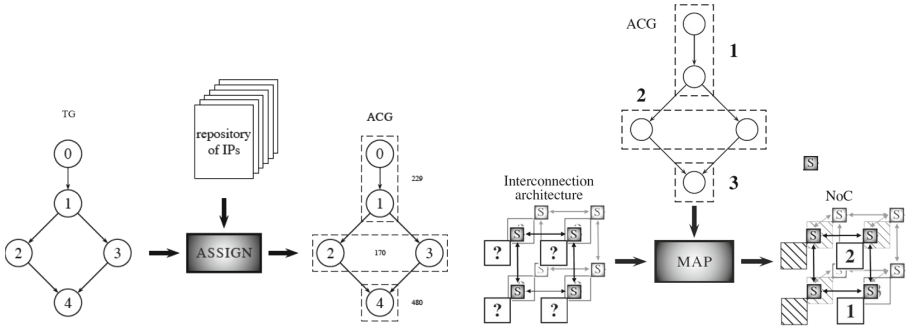
4 IP Assignment and Mapping

The platform-based design methodology for SoC encourages the reuse of components to reduce costs and to reduce the time-to market of new designs. The designer of NoC-based systems faces two main problems: selecting the adequate set of IPs and finding the best physical mapping of these IPs into the NoC structure. On a platform-based design, the selection of IPs is called IP assignment stage and the physical mapping is called IP mapping stage.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<repository>
  <ips>
    <ip procName="AMD_ElanSC520-133_MHz--square"
     price="33.0" taskTime="9e-06" taskPower="1.6"
     area="9.61E-6" taskName="Angle2Time Conversion"
     type="0" procID="0" id="0"
    />
    <ip procName="AMD_ElanSC520-133_MHz--square"
     price="33.0" taskTime="2.3e-05" taskPower="1.6"
     area="9.61E-6" taskName="Basic floating point"
     type="1" procID="0" id="1"
    />
    <ip procName="AMD_ElanSC520-133_MHz--square"
     price="33.0" taskTime="0.00049"taskPower="1.6"
     area="9.61E-6" taskName="Bit Manipulation"
     type="2" procID="0"id="2"
    />
     . . .
  </ips>
</repository>
```

Fig. 3. XML code used to represent the repository

IP Assignment is the first step before mapping the application onto NoC [19]. The objective of IP Assignment is to select, from an IP library (IP repository), a set of IPs that exploit re-usability and optimize the implementation of a given application in terms of time, power and area requirements. During this step, no information about physical location of IPs onto NoC is given. The optimization process must be done based on TG and IP features only. The result of this step is a set of IPs that should maximize the NoC performance, *i.e.* minimize power consumption, hardware resources as well as the total time execution of the application. Recall that the result of the assignment step is produced in the form of an ACG for the application's task graph, wherein each task has an IP associated with it. The dynamics of this assignment step is illustrated in Fig. 4(a).



(a) Dynamics of the assignment process (b) Process of the IP mapping problem

Fig. 4. IP assignment and mapping problems

Given an application, the problem that we are concerned with here is to determine how to topologically map the selected IPs onto the network structure, such that the objectives of interest are optimized. At this stage, a more accurate evaluation can be done taking into account the distance between resources and the number of switches and channels crossed by a data package during a communication session. The result of this process should be an optimal allocation of the prescribed IP assignment to execute the application on the NoC.

5 IP Mapping with Multi-objective Particle Swarm Algorithm

The successful application of PSO in many single objective optimization problems reflects the effectiveness and robustness of PSO. The PSO algorithm for single objective as proposed in [23] is a population-based search algorithm that simulates the social behavior of birds within a flock: the particle simulates the bird's movement while the particle swarm mimics that of the flock. In PSO, each particle position represents a potential solution of the problem.

The particle position x_i, which is a vector of as many coordinates as the dimensions of the search space. It is updated every iteration t using the velocity information as described in Eq. 1:

$$x_i(t+1) = x_i(t) + v_i(t+1). \tag{1}$$

The velocity drives the optimization process to the next position. It is computed as described in Eq. 2:

$$v_i(t+1) = w.v_i(t) + c_1 \times r_1(t)(Pbest(t) - x_i(t)) + c_2 \times r_2(t)(Gbest(t) - x_i(t)). \tag{2}$$

Three terms that are needed to compute the velocity: (i) the inertia, which is based on the previous velocity $w.v_i(t)$. In this term w is the inertia coefficient; (ii) the cognitive term $c_1.r_1(t)(Pbest(t) - x_i(t))$, which quantifies the performance of particle i with respect its previous performance. This component is also known as the "nostalgia" of the particle. In this term, c_1 is the cognitive coefficient and $Pbest(t)$ is the best position founded by particle i so far; (iii) the social component, $c_2.r_2(t)(Gbest(t) - x_i(t))$ which quantifies the performance of particle i with respect to the performance of the swarm of particles. The effect of this term is to attract the particle to the best position found by the particles swarm $Gbest(t)$. In this term, coefficient c_2 is the social coefficient. The cognitive coefficient c_1 and the social coefficient c_2 yield a better performance when they are balanced [24]. Note that $r_1(t)$ And $r_2(t)$ are random numbers selected in the range of [0, 1] at each iteration t.

The PSO algorithm cannot be immediately applied to multi-objective optimization problem. In single-objective optimization, the best solution that each particle uses to update its position is completely determined. However, in multi-objective optimization problems, each particle might have a set of different best, also known as leader, from which just one has be selected in order to update the

particle's position. Such set of leaders is usually stored in a different memory, which is distinct of that of the swarm. This memory is identified as the external archive. This is a repository in which the non-dominated solutions found so far are stored. The solutions contained in the external archive are used as leaders when the positions of the particles of the swarm have to be updated. Furthermore, the contents of the external archive are also usually reported as the final results of the algorithm.

In this paper, we improved the PSO method to multi-objective approach. The important part in multi-objective particle swarm optimization (MOPSO) is to determine the best global particle for each particle i of the swarm. To facilitate the selection of best and local best, two archives are maintained in addition to the current state of the swarm. One archive stores the global best individuals found so far by the search process and the other bookkeep the set of local best positions for each member of the swarm. in our approach, the selection is random. The main steps of the algorithm of the applied MOPSO are shown in Algorithm 1.

Algorithm 1. The main steps of modified MOPSO

```
Initialize the particle of the swarm
Evaluate all the particles of the swarm
Initialize best solutions in archive of leaders
Initialize the archive p_Leader
Initialize the velocity for each particle.
iteration := 0
while iteration < max_iteration do
  for each particle do
    Select a p_leader
    Update Position using Eqs 1 and 2
    Evaluate particle fitness
    Update p_Leader
  end for
  Update the leaders archive
  iteration := iteration + 1
end while
Return result from the archive of leaders
```

The PSO is usually used for the optimization of continuous function. However, in this work we use a discrete version of the multi-objective PSO.

6 Objective Functions

Different objective may be considered in the IP assignment and mapping problems. The objectives can be concurrent or collaborative [4]. Concurrent objective should not be grouped, and considered separately during optimization process. Thus, the process can be treated as a multi-objective optimization.

The best solution for multi-objective optimization is the solution with the adequate trade-off between all concurrent objectives. In this paper, we adopt a multi-objective optimization minimizing three objectives: area, power consumption and execution time. These objectives must be computed.

6.1 Area

In order to compute the area required by a given mapping, it is necessary to know the area needed for the selected processors and that required by the used links and switches. As a processor can be responsible for more than one task, each APG node must be visited in order to check the processor identification in the appropriate XML element. Grouping the nodes with the same *processorID* attribute allows us to implement this verification. The total number of links and switches can be obtained through the consideration of all communication paths between exploited tiles. Note that a given IP mapping may not use all the available tiles, links and switches. Also, observe that a portion of a path may be re-used in several communication paths.

Among the routing strategy exists in the literature [25], in this work we adopted a XYZ fixed routing strategy, wherein data emanating from tile i is sent first to the West or East side of the corresponding switch, depending on the target tile position, say j, with respect to i in the NoC 3D mesh, until it reaches the column of tile j. Then, it is sent to the south or North side, also depending on the position of tile j with respect to tile i. Finally, it is sent to the Top or Bottom side until it reaches the target tile. Each communication path between tiles is stored in the routing table. The number of links in the aforementioned route can be computed as described in Eq. 3. This also represents the distance between tiles i and j and it is called the *Manhattan distance* [25].

$$nLinks(i,j) = |x_i - x_j| + |y_i - y_j| + |z_i - z_j| \tag{3}$$

wherein (x_i, y_i, z_i) and (x_j, y_j, z_j) are the 3-dimensional coordinate of tiles i and j respectively.

In the purpose of computing efficiently the area required by all used links and switches, an APG can be associated with a so-called *routing table* whose entries describe appropriately the links and switches necessary to reach a tile from another. The number of hops between tiles along a given path leads to the number of links between those tiles, and incrementing that number by 1 yields the number of traversed switches. The area is computed summing up the areas required by the implementation of all distinct processors, switches and links.

Equation 4 describes the computation involved to obtain the total area for the implementation a given IP mapping M:

$$Area(M) = \sum_{p \in Proc(APG_M)} area_p + (A_l + A_s) \times Links(APG_M) + A_s \tag{4}$$

wherein function $Proc(.)$ provides the set of distinct processors used in APG_M and $area_p$ is the required area for processor p, Function $Links(.)$ gives the number of distinct links used in APG_M and A_l is the area of any given link and A_s is the area of any given switch.

6.2 Power Consumption

The total power consumption of an application NoC-based implementation consists of the power consumption of the processors while processing the computation performed by each IP and that due to the data transportation between the tiles is presented in Eq. 5:

$$Power(M) = Power_p(M) + Power_c(M) \tag{5}$$

wherein $Power_p(M)$ and $Power_c(M)$ are the processing and communication power consumption, respectively.

The former can be computed summing up attribute *taskPower* of all nodes of the APG and the latter is the power consumption due to communication between the application tasks through links and switches. The power consumption due to the computational activity is simply obtained summing up attribute *taskPower* of all nodes in the APG and is as described in Eq. 6:

$$Power_p(S) = \sum_{t \in APG_S} power_t \tag{6}$$

The power consumed due communication is an important feature to be considered on a NoC power model in order to get an accurate evaluation. This feature depends on the application communication pattern and the NoC platform. The communication pattern is given by the assignment and mapping, while the NoC platform is defined by the network topology, switching strategy and routing algorithm. The total power consumption of sending one bit of data from tile i to tile j can be calculated considering the number of switches and links the bit passes through on its way along the path. It can be estimated as shown in Eq. 11:

$$E_{bit}^{i,j} = nLinks(i,j) \times E_{L_{bit}} + (nLinks(i,j) + 1) \times E_{S_{bit}} \tag{7}$$

wherein $E_{S_{bit}}$ and $E_{L_{bit}}$ represent the energy consumed by the switch and link tying the two neighboring tiles, respectively [1]. Function $nLinks(.)$ provides the number of traversed links (and switches too) considering the routing strategy used in this work and described earlier in this section. The function is defined in Eq. 3.

Recall that the application TG gives the communication volume ($V(d_{t,t'})$) in terms of number of bits sent from the task t to task t' passing through a direct arc $d_{t,t'}$. Assuming that the tasks t and t' have been mapped onto tiles i and j respectively. The total network communication power consumption for a given mapping M is given in Eq. 8, wherein $Targets_t$ provides all tasks that have a

direct dependency on data resulted from task t and $Tile_t$ yields the tile number into which task t is mapped.

$$Power_c(M) = \sum_{\substack{t \in APG_M, \\ \forall t' \in Targets_t}} V(d_{t,t'}) \times E_{bit}^{Tile_t, Tile_{t'}} \tag{8}$$

6.3 Execution Time

To compute the execution time of a given mapping, we consider the execution time of each task, their schedule and the additional time due to data transportation through links and switches along the communication path. The execution time of each task is defined by the *taskTime* attribute in TG. It is necessary to visit all the tasks of the its TG and schedule them into the own cycle. Thus, a scheduling algorithm ought to be applied.

This paper presents two kinds of scheduling algorithm [26]: one based on the As-Soon-As-Possible (ASAP) scheduling strategy and the As-Late-As-Possible (ALAP) scheduling strategy. The first algorithm schedules tasks in the earliest possible control step. The second algorithm schedules tasks in the latest possible control step.

Links and switches can be counted using the routing table. We identified two situations that can degrade the implementation performance, increasing the execution time of the application:

1. *Parallel tasks with partially shared communication path:*
   When a task in a tile must send data to supposedly parallel tasks in different tiles through the same *initial* link, data to both tiles cannot be sent at the same time.
2. *Parallel tasks with common target using the same communication path:*
   When several tasks need to send data to a common target task, one or more shared links along the partially shared path would be needed simultaneously. The data from both tasks must then be pipelined and so will not arrive at the same time to the target task.

Equation 9 gives the execution time considering the computation and communication for a given mapping solution M:

$$Time(M) = Time_S + Time_c + t_L \times (F_1 + F_2) \tag{9}$$

where $Time_S$ returns the time necessary to processes the tasks of The TG and $Time_c$ the time spent due communication among tasks. It is necessary to add the delay concerning the last two aforementioned situations. Delay caused by the first situation is computed by function F_1 and delay caused the second situation is computed by function F_2.

Function F_1 computes the additional time due to parallel tasks that have data dependencies on tasks mapped in the same source tile and yet these share a common initial link in the communication path. Function F_2 computes the

additional time due to the fact that parallel tasks producing data for the same target task need to use simultaneously at least a common link along the communication path.

The time spent due communication between tasks is computed as shown in Eq. 10, which is a extension of the time model presented in [27]:

$$Time_c = \sum_{\substack{t \in APG_M, \\ \forall t' \in Targets_t}} \left\lceil \frac{V(d_{t,t'})}{phit} \right\rceil * T^{t,}_{phit} \tag{10}$$

wherein $V(d_{t,t'})$ is the volume of bits transmitted from task t to task t' and the time required to transfer a *phit* is defined as in Eq. 11:

$$T^{t,t'}_{phit} = nLinks(t,t') \times T_{L_{phit}} + (nLinks(t,t') + 1) \times T_{S_{phit}} \tag{11}$$

where $T_{L_{phit}}$ is the link transmission time and $T_{S_{phit}}$ is the switch processing time used to transfer one phit between two neighboring tiles.

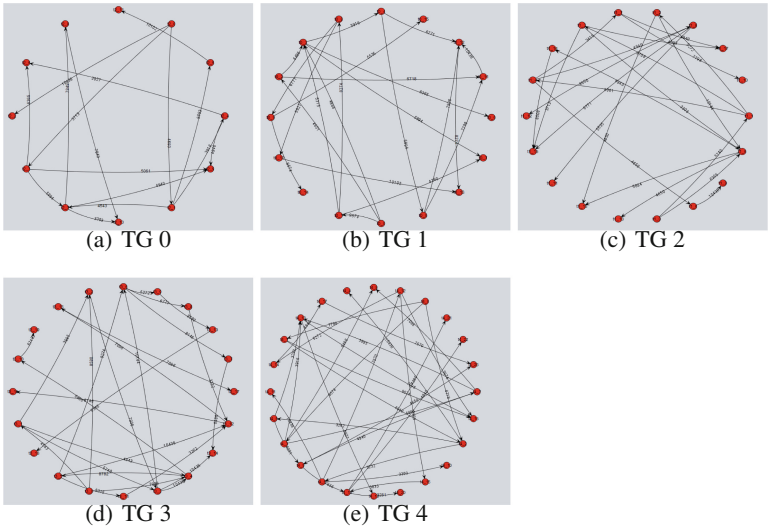
7 Performance Results

The E3S benchmarks suite is used to evaluate the proposed approach. The suite contains the characteristic of 17 embedded processors. These processors are characterized by the measured execution times of 16 different types of tasks, power consumption derived from processor data-sheets, die size required, price and clock frequency. These applications are described by the corresponding TGs. Each one of the nodes in these TGs is associated with a task type. A task type is a processor instruction or a set of instructions. If a given processor is able to execute a given type of instruction, so that processor is a candidate to be mapped onto a resource in the NoC structure and would be responsible for the execution of one or more tasks of the TG. We use 5 random task graphs, generated by the TGFF [5] to perform experiments and evaluate the performance. The generated applications have the characteristics given in Table 1, and they task graphs illustrated in Fig. 5.

Several tentative simulations were performed to find out the set of parameters setting to be in the proposed algorithm. We exploit a population size of

Table 1. Characteristics of the applications

App Id	*#Tasks*	*#Arcs*
TG 0	12	16
TG 1	16	22
TG 2	18	23
TG 3	20	27
TG 4	24	32



(a) TG 0 (b) TG 1 (c) TG 2

(d) TG 3 (e) TG 4

Fig. 5. The task graphs of the mapped applications as generated by the TGFF

500 with w set to 0.8 and c_1 and c_2 both set to 1.49. The algorithm was run for 1000 iterations. Moreover, two scheduling algorithms are used in the simulation: ASAP, whereby a given task is scheduled As Soon As Possible and ALAP whereby a given task is scheduled As Late As Possible.

We provide different results for these applications when using ASAP and ALAP scheduling algorithm. Tables 2 and 3 show the characteristics of the found Mappings for the ASAP scheduling strategy, while Tables 4 and 5 show the same characteristics of the found Mappings for the ALAP scheduling strategy. For each of these applications, we show the total number of non-dominated mapping, the best and the average of power consumption, the hardware area and the execution time. These values are not necessarily extracted from the same solution. Note that the power figures are *Watts*, time is 10^{-3} s and area is 10^{-6} m^2.

Table 2. Results regarding IP mapping in 5×5 NoC with ASAP

App ID	#Mapping	Best			Average		
		Power	Time	Area	Power	Time	Area
0	24	522.66	5.69	59.68	623.83	32.20	129.22
1	20	957.94	49.51	98.32	1107.86	116.17	233.77
2	34	819.11	9.56	134.97	960.25	247.58	204.30
3	34	1159.09	42.06	135.62	1390.71	64.43	229.94
4	34	1149.61	24.96	218.76	1379.34	34.86	415.20

Table 3. Results regarding IP mapping in $3 \times 3 \times 3$ NoC with ASAP

App ID	#Mapping	Best			Average		
		Power	Time	Area	Power	Time	Area
0	43	403.25	5.27	45.11	479.44	6.37	113.48
1	58	757.64	47.95	77.16	855.37	135.07	161.62
2	66	687.44	8.57	116.74	819.26	202.03	213.46
3	47	953.52	40.88	123.21	1124.44	49.48	214.21
4	88	1017.02	24.19	147.50	1149.18	129.50	338.37

Table 4. Results regarding IP mapping in 5×5 NoC with ALAP

App ID	#Mapping	Best			Average		
		Power	Time	Area	Power	Time	Area
0	12	581.90	5.86	64.96	644.48	7.18	179.56
1	28	853.76	51.44	106.57	1001.96	117.29	207.70
2	32	898.65	10.21	135.79	995.52	38.90	220.25
3	25	1103.91	41.70	125.89	1288.34	159.33	221.51
4	41	1189.49	25.55	191.10	1397.00	47.75	347.90

Besides comparing the scheduling algorithm, two topologies are used in the test 2D and 3D mesh. Tables 2 and 4 show the results obtained using 2D mesh, while Tables 3 and 5 show the results obtained using 3D mesh. When we compare the two scheduling strategies in 2D mesh, the comparison between Tables 2 and 4 shows that the results achieved by ASAP scheduling are a better than those evolved by ALAP scheduling. Also, the results achieved by ASAP scheduling are superiors to those evolved by ALAP scheduling in 3D mesh, as shown in Table 5. On the hand, when we compare the two NoCs topologies, it is safe to conclude that mapping in 3D mesh achieves better results than 2D mesh because of the vast number of non-dominated solutions discovered by 3D mesh when compared to those yield by 2D mesh.

Table 5. Results regarding IP mapping in $3 \times 3 \times 3$ NoC with ALAP

App ID	#Mapping	Best			Average		
		Power	Time	Area	Power	Time	Area
0	59	451.18	5.44	50.79	526.68	49.08	137.10
1	53	742.42	51.38	83.74	833.68	335.12	194.59
2	121	691.89	9.34	112.14	846.44	156.93	248.33
3	80	976.84	41.68	127.94	1146.13	163.81	231.15
4	101	1021.83	25.72	219.17	1189.82	49.99	391.36

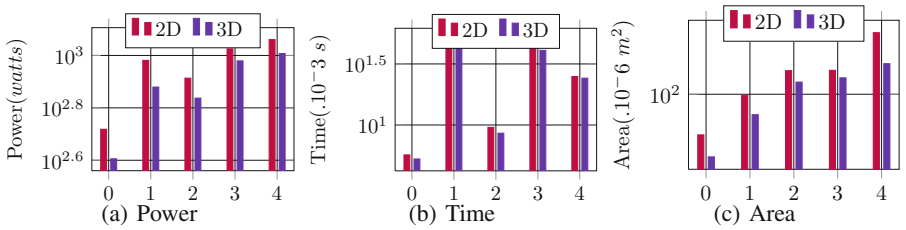


Fig. 6. IP mapping comparison using ASAP scheduling

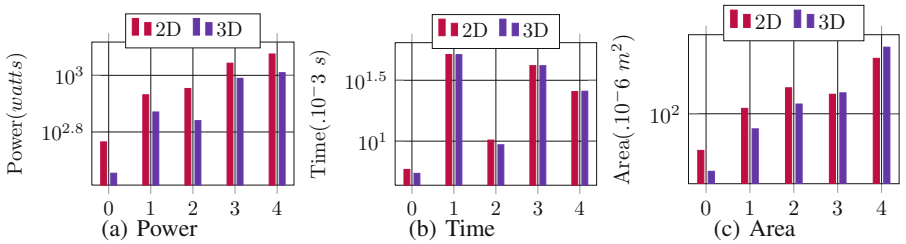


Fig. 7. IP mapping comparison using ALAP scheduling

Besides comparing the solutions obtained, the charts of Figs. 6(a) and 7(a) allow a visual comparison of the power consumption for the mapping yield by the compared topology, regarding the best results. Also, the charts of Figs. 6(b) and 7(b) give a better visual comparison of results regarding the time characteristic considering the best. Finally, the charts of Figs. 6(c) and 7(c) facilitate the comparison of the required hardware area that would be due to the mapping obtained by the compared topologies.

8 Conclusions

The problem of assigning and mapping IPs are NP-hard problems in the NoC design. In this paper, we propose a multi-objective algorithm based on Particle Swarm Optimization to help NoC designers to select a set of appropriate IP addresses from the IP repository and mapping a prescribed set of IPs into an NoC physical structure. We use TGFF [5] to generate randomly the task graphs, and we adopt the E3S benchmark as the IPs benchmark. We use the same objectives used in [4]. Two scheduling algorithm has been tested in our tool. The obtained results show that using 3D mesh topology perform better result than using 2D mesh, The obtained results show also that ASAP scheduling strategy is better then ALAP scheduling. For future work, we intend to use more routing strategies. We also intend to apply other new evolutionary techniques.

References

1. Hu, J., Marculescu, R.: Energy-aware mapping for tile-based NoC architectures under performance constraints. In: Proceedings of the 2003 Asia and South Pacific Design Automation Conference, pp. 233–239. ACM (2003)
2. Ogras, U.Y., Hu, J., Marculescu, R.: Key research problems in NoC design: a holistic perspective. In: Proceedings of the 3rd IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis, pp. 69–74. ACM (2005)
3. Gary, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman & Co., New York (1979)
4. Da Silva, M.V.C., Nedjah, N., de Macedo Mourelle, L.: Evolutionary IP assignment for efficient NoC-based system design using multi-objective optimization. In: 2009 IEEE Congress on Evolutionary Computation, pp. 2257–2264. IEEE (2009)
5. Dick, R.P., Rhodes, D.L., Wolf, W.: TGFF: task graphs for free. In: Proceedings of the Sixth International Workshop on Hardware/Software Codesign (CODES/CASHE 1998), pp. 97–101. IEEE (1998)
6. Reyes-Sierra, M., Coello Coello, C.A., et al.: Multi-objective particle swarm optimizers: a survey of the state of-the-art. Int. J. Comput. Intell. Res. **2**(3), 287–308 (2006)
7. He, H., Fang, F., Wang, W.: Improved simulated annealing genetic algorithm based low power mapping for 3d NoC. In: MATEC Web of Conferences, vol. 232, p. 02022. EDP Sciences (2018)
8. Li, J., et al.: Bat algorithm based low power mapping methods for 3D network-on-chips. In: Du, D., Li, L., Zhu, E., He, K. (eds.) NCTCS 2017. CCIS, vol. 768, pp. 277–295. Springer, Singapore (2017). https://doi.org/10.1007/978-981-10-6893-5_21
9. Huang, C., Zhang, D., Song, G.: A novel mapping algorithm for three-dimensional network on chip based on quantum-behaved particle swarm optimization. Front. Comput. Sci. **11**(4), 622–631 (2017)
10. Huang, C., Zhang, D., Song, G.: Low-power mapping algorithm for three-dimensional network-on-chip based on diversity-controlled quantum-behaved particle swarm optimization. J. Algorithms Comput. Technol. **10**(3), 176–186 (2016)
11. Mosayyebzadeh, A., Amiraski, M.M., Hessabi, S.: Thermal and power aware task mapping on 3d network on chip. Comput. Electr. Eng. **51**, 157–167 (2016)
12. Wadhwani, P., Choudhary, N., Singh, D.: Energy efficient mapping in 3d mesh communication architecture for NoC. Glob. J. Comput. Sci. Technol. (2013)
13. Addo-Quaye, C.: Thermal-aware mapping and placement for 3-d NoC designs. In: Proceedings 2005 IEEE International SOC Conference, pp. 25–28. IEEE (2005)
14. Wang, J., Li, L., Pan, H., He, S., Zhang, R.: Latency-aware mapping for 3d NoC using rank-based multi-objective genetic algorithm. In: 2011 9th IEEE International Conference on ASIC, pp. 413–416. IEEE (2011)
15. Sepulveda, J., Gogniat, G., Pires, R., Chau, W., Strum, M.: An evolutive approach for designing thermal and performance-aware heterogeneous 3D-NoCs. In: 2013 26th Symposium on Integrated Circuits and Systems Design (SBCCI), pp. 1–6. IEEE (2013)
16. Bhardwaj, K., Mane, P.S.: C3Map and ARPSO based mapping algorithms for energy-efficient regular 3-D NoC architectures. In: Technical papers of 2014 International Symposium on VLSI Design, Automation and Test, pp. 1–4. IEEE (2014)

17. Elmiligi, H., Gebali, F., El-Kharashi, M.W.: Power-aware mapping for 3D-NoC designs using genetic algorithms. Procedia Comput. Sci. **34**, 538–543 (2014)
18. Davis, W.R., et al.: Demystifying 3D ICS: the pros and cons of going vertical. IEEE Des. Test Comput. **22**(6), 498–510 (2005)
19. Bougherara, M., Nedjah, N., De Macedo, L., Mourelle, R.R., Sadok, A., Bennouar, D.: IP assignment for efficient NoC-based system design using multi-objective particle swarm optimisation. Int. J. Bio-Inspired Comput. **12**(4), 203–213 (2018)
20. Bougherara, M., Nedjah, N., Bennouar, D., Rahmoun, R., Sadok, A., de macedo Mourelle, L.: Core/task associations for efficient application implementation on network-on-chip. In: 2018 International Conference on Computer and Applications (ICCA), pp. 18–22. IEEE (2018)
21. Bougherara, M., Kemcha, R., Nedjah, N., Bennouar, D., de macedo Mourelle, L.: IP assignment optimization for an efficient NoC-based system using multi-objective differential evolution. In: International Conference on Metaheuristics and Nature Inspired Computing (META), pp. 435–444 (2018)
22. Dick, R.P.: Embedded System Synthesis Benchmarks Suite (E3S). http://ziyang.eecs.northwestern.edu/dickrp/e3s/
23. Zhang, L.-B., Zhou, C.-G., Liu, X.H., Ma, Z.Q., Ma, M., Liang. Y.C.: Solving multi objective optimization problems using particle swarm optimization. In: The 2003 Congress on Evolutionary Computation, CEC 2003, vol. 4, pp. 2400–2405. IEEE (2003)
24. Nedjah, N., de Macedo Mourelle, L.: Evolutionary multi-objective optimisation: a survey. Int. J. Bio-Inspired Comput. **7**(1), 1–25 (2015)
25. Liu, S., et al.: Freerider: non-local adaptive network-on-chip routing with packet-carried propagation of congestion information. IEEE Trans. Parallel Distrib. Syst. **26**(8), 2272–2285 (2015)
26. Gruian, F., Kuchcinski, K.: LEneS: task scheduling for low-energy systems using variable supply voltage processors. In: Proceedings of the 2001 Asia and South Pacific Design Automation Conference, pp. 449–455. ACM (2001)
27. Kreutz, M., Marcon, C.A., Carro, L., Wagner, F., Susin, A.A.: Design space exploration comparing homogeneous and heterogeneous network-on-chip architectures. In: Proceedings of the 18th Annual Symposium on Integrated Circuits and System Design, pp. 190–195. ACM (2005)