Parallel Applications Mapping onto Heterogeneous MPSoCs interconnected using Network on Chip

Dihia Belkacemi, Daoui Mehammed, Samia Bouzefrane

Parallel Applications Mapping onto Heterogeneous MPSoCs interconnected using Network on Chip

Dihia Belkacemi[1], Mehammed Daoui[1], and Samia Bouzefrane[2]

[1] Laboratoire de Recherche en Informatique, Université de Tizi-Ouzou, Algeria
[2] CEDRIC Lab, CNAM, France
{samia.bouzefrane}@cnam.fr

Abstract. To meet the growing requirements of today's applications, multiprocessor architectures (MPSoCs) interconnected with a network on chip (NoC) are considered as a major solution for future powerful embedded systems. Mapping phase is one of the most critical challenge in designing these systems. It consists of assigning application' tasks on the target platform which can have a considerable influence on the performance of the final system. Due to the large solutions' research space generated by both the application complexity and the platforms, this mapping phase can no longer be done manually and hence it requires powerful exploration tools called DSE (Design Space Exploration Environment). This paper proposes a new tool for static mapping applications on NoC based on heterogeneous MPSoCs. This tool integrates several multiobjective optimization algorithms that can be specified in order to explore different solutions' spaces, mainly: exact method, metaheuristics (population-based metaheuristics and single solution-based ones) as well as hybrid ones; it offers different cost functions (defined using analytical or simulation models). The user can specify them or define others easily and it provides an easy way to evaluate the performance of the Pareto front returned by different algorithms using multiple quality indicators. We also present a series of experiments by considering several scenarios and give guidelines to designers on choosing the appropriate algorithm based on the characteristics of the mapping problem considered.

Keywords: Static Mapping · Multiobjective Optimization · Network on Chip (NoC)· Multi-processor System on Chip (MPSoCs).

1 Introduction

In order to meet today's applications requirements (e.g. multimedia, digital signal processing, image processing, etc.), MPSoCs are becoming increasingly popular solutions for future embedded systems [1]. These MPSoCs are classified as homogeneous or heterogeneous. Several studies have shown that heterogeneous MPSoCs outperform their homogeneous counterparts [2][3]. Heterogeneous MPSoCs are composed of PEs of different types such as General Purpose Processor

(GPP), Application Specific Integrated Circuit (ASIC), Digital Signal Processor (DSP), hardware accelerators or IP blocks (e.g. video encoder), etc. The distinct features of different processing elements (PEs) provide high performance with less energy consumption. Typical examples of heterogeneous MPSoCs are ST Nomadik for cellular phones [4], NVIDIA's Tegra designed for multimedia in mobile phones [5], Samsung Exynos 5422 SoC [6], etc... P The growing complexity of heterogeneous MPSoCs requires effcient communication infrastructure. Network on Chip (NoC) [7] is used as the communication infrastructure which provides modularity and scalability unlike buses and point to point communication. A NoC consists of a set of processing elements (PEs), routers, links, and network interfaces (NIs) as shown in Figure 1. Each PE is connected to a router and accesses to the network through NI. Routers are interconnected using a set of links to form a given topology (e.g. 2D Mesh, 2D Torus, etc...). The processing elements (PEs) interconnected to the NoC exchange data in the form of packets (set of flits). Routing and switching policies are used respectively to determine the path and how the data is communicated between PEs (source and destination). NoCs are also characterized by arbitration as well as flow control techniques. Arbitration is used to solve contentions and flow control technique defines how NoC resources are allocated to data.

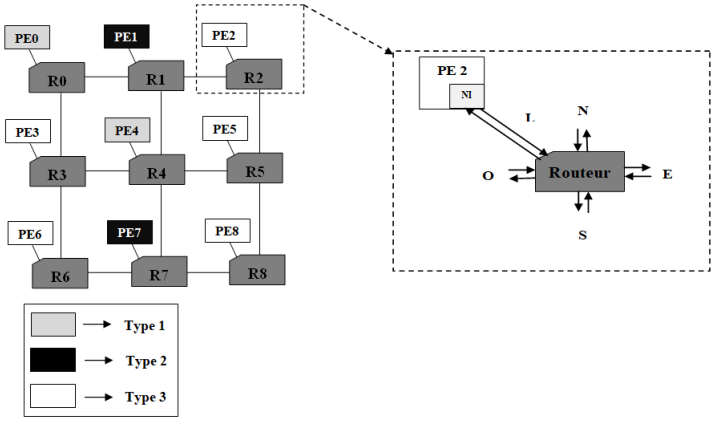

Fig. 1. Example of Heterogeneous MPSoC with three types of processors interconnected using 2D Mesh NoC Topology

One critical problem of heterogeneous MPSoCs interconnected using Network on Chip is how to map an application on this platform which is a NP-hard [8] problem. In addition, when solving the mapping problem, the designer is often called to specify several objectives to be optimized simultaneously. These objectives are often conflicting (e.g. communication vs load, energy consumption vs total execution time, etc.), so the goal is to find reliable trade-off mapping solutions which simultaneously optimize the different contradictory metrics (ob-

jectives) commonly called Pareto Optimal Solutions Set (i.e. Mapping problem is an instance of a multiobjective optimization problem MOP). To tackle all these challenges, designing tools that can automate this step is required. This paper presents a new tool for mapping applications on NoC based on heterogeneous MPSoCs. The work reported in this article extends our prior work [9] with the following contributions:

- In addition to the analytical models used to define the cost (or objective) functions, we propose a simulation model to take into account the dynamic aspect (like contentions) of the system during the mapping process.

- In order to enrich our mapping tool, we have added other metaheuristics like single-solution based metaheuristics (Archived Multiobjective Simulated Annealing (AMOSA) [10] and Parallel Multiobjective Tabu Search (PMOTS) [11]) and hybrid ones.

- We have implemented multiobjective exact method called Multiobjective Branch and Bound (MBB) [12] to check the efficiency of the metaheuristics presented in the tool in solving small mapping problem instances.

The remainder of this paper is organized as follows. Section 2 describes the multiobjective optimization principles. The related works are discussed in Section 3. In Section 4, the proposed exploration tool is presented. Experimental results are presented and discussed in Section 5. Section 6 concludes this paper.

2 Multiobjective Optimization

A multiobjective optimization problem (MOP) can be mathematically formulated as given by [22]:

$$MOP = \begin{cases} "min" F(x) = (f_1(x), \ldots, f_n(x)) \\ \\ x \in S \end{cases}$$

where n ($n \geq 2$) is the number of objectives, $x = (x_1, \ldots, x_k)$ is the vector representing the decision variables, and S represents the set of feasible solutions associated with equality and inequality constraints and explicit bounds. $F(x) = (f_1(x), \ldots, f_n(x))$ is the vector of objectives to be optimized. The objectives in MOP are often conflicting. Before defining a Pareto optimality, a partial order relation could be defined, known as dominance relation.

Definition 1. ***Pareto dominance [22].*** *An objective vector $u = (u_1, \ldots, u_n)$ is said to dominate another vector $v = (v_1, \ldots, v_n)$, denoted as $u \prec v$, if and only if no component of v is smaller than the corresponding component of u and at least one component of u is strictly smaller, that is,*

$\forall i \in \{1, \ldots, n\}, u_i \leq v_i$ *and* $\exists i \in \{1, \ldots, n\} : u_i < v_i$.

Definition 2. ***Pareto optimality [22].*** *A solution $x^* \in S$ is Pareto optimal if for every $x \in S$, $F(x)$ does not dominate $F(x^*)$, that is, $F(x) \nprec F(x^*)$.*

Definition 3. ***Pareto optimal set [22].*** *For a given $MOP(F, S)$, the Pareto optimal set is defined as $P^* = \{x \in S / \nexists x^* \in S, F(x^*) \prec F(x)\}$.*

Definition 4. ***Pareto front [22].*** *For a given $MOP(F, S)$ and its Pareto optimal set P^*, the Pareto front is defined as $PF^* = \{F(x), x \in P^*\}$.*

3 Related Work

Several authors proposed to use multiobjective optimization algorithms to solve the mapping problem [13–21]. Ascia et al. [13] present an approach based on SPEA2 metaheuristic for exploring the mapping design space. Their aim is to obtain the Pareto mappings that maximize performance and minimize the amount of power consumption. Erbas et al. [14] give a comparative study between two metaheuristics NSGAII and SPEA2. Their aim is to optimize processing time, power consumption, and architecture cost. Zhou et al. [15] address the problem of topological mapping of Intellectual Properties (IPs) on the tile of a mesh-based NoC using NSGA metaheuristic while treating two conflicting objectives: minimizing the average hop and achieving the thermal balance. A multiobjective genetic algorithms MOGA to determine the Pareto-optimal configuration which optimizes average delay and routing robustness is presented in [16]. In [17], authors propose the use of multiobjective evolutionary algorithms (NSGAII and MicroGA) to minimize hardware area, execution time and the total power consumption. Wu et al. [18] propose a new mapping algorithm based on the Genetic Algorithm (GA) and the MAX-MIN Ant System Algorithm (MMAS) called GA-MMAS to optimize power consumption and NoC latency. He and Guo [19] use ACO to solve the mapping problem while optimizing communication power consumption and delay. Chatterjee et al. [20] propose a constructive heuristic method to solve the problem of mapping applications on a NoC (with a mesh topology). Their goal is to optimize the cost of network communications as well as the reliability of the system. Bruch et al. [21] present an optimization flow for mapping applications on a NoC in order to meet the time requirements and minimize the costs of using virtual channels. The approach used is based on the NSGAII algorithm. Authors in [18–20] used the aggregation approach (using a unified cost function) in order to take several objectives into account during the mapping. The disadvantage of their approach comes from the difficulty to adjust the weights which requires knowledge of the problem. Other works like [15, 17] did not take into account the dynamic effects (i.e. contentions) of the NoC during mapping. An important limitation of these approaches is that only a few metaheuristics like NSGA ([15]), NSGAII ([14, 17, 21]), SPEA2 ([13, 14]), MOGA ([16]), etc. are mainly explored. Most of the works take into account the two-dimensional optimization space (i.e. optimize only two cost functions), e.g. performance and energy consumption in [13], minimize the average number of hops and achieves a thermal balance [15], etc.

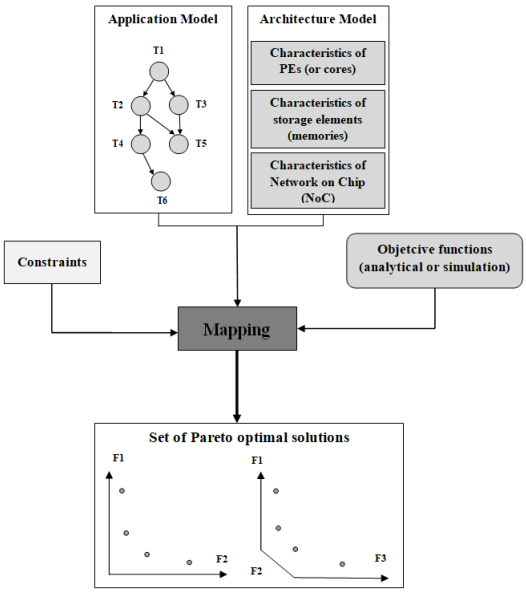

Fig. 2. Inputs and Outputs of the proposed mapping tool

4 Description of the proposed tool

Figure 2 gives an overview of the proposed mapping tool. This tool has as inputs (1) the application model represented as an annotated task graph, (2) the high level architecture model, (3) the objective functions defined using analytical or simulation models and (4) the architecture and application constraints. A set of multiobjective optimization algorithms is used to explore the mapping space and to find a set of Pareto optimal solutions.

4.1 Application model

The application model is represented in the form of a Directed Acyclic Graph (DAG), denoted $G(V, E)$. The set of nodes V=$\{T_0$,T_1, ...,$T_n\}$ represents the application's tasks and E is a set of edges e_i. Each edge e_i in E designates the precedence relation between two tasks connected by e_i labeled with $volume(e_i)$ representing the amount of data exchanged between these tasks. Each task T_i is annotated with $Load(T_i)$ which is the number of instructions of task T_i. Two vectors Ei and Ci contain respectively the energy consumption and execution time of task T_i on each type of processing element. A task T_i may have a deadline $d(T_i)$.

4.2 Architecture model

The architecture model includes the high level description of the target platform that will run the application. The informations considered in our case are:

- The characteristics of the processors (e.g. number, type, frequency, etc.)
- The characteristics of the storage elements (e.g. maximum memory capacity, etc.) and
- NoC features including: router characteristics (e.g. routing algorithm, switching modes, flow control techniques, arbitration, etc.); the characteristics of the links (e.g. direction, link rate, etc.) and the NoC's topology.

4.3 Objective functions

We have defined objective functions using two models: analytical and simulation.

Analytical model It consists in finding mathematical equations used to evaluate a solution of a given mapping. This model has the advantage of being less expensive in terms of execution time at the expense of the level of precision. In this section, cost functions which measures the quality of a given mapping using analytical model is presented. In the rest of this paper, we use a decision variable $x_{i,j}$ which is defined as follows:

$$x_{ij} = \begin{cases} 1 \text{ if } T_i \text{ is assigned to processor } PE_j \\ 0 \text{ otherwise} \end{cases}$$

Load balancing This cost function gives the load balancing between the different PEs for a given mapping, such that all the processing elements in the system are equally loaded (avoiding task concentration in just some processors). This cost function is defined as follows:

$$\sum_{j=0}^{P-1} abs\left(\frac{load(PE_j)}{f(PE_j)} - M\right) \tag{1}$$

where *load(PE_j)* represents the workload of the processor PE_j expressed as the sum of instructions of tasks that run on it.

$$load(PE_j) = \sum_{Ti \in V} x_{ij} \times load(T_i) \tag{2}$$

M represents the average load.

$$M = \frac{\sum_{j=0}^{P-1} load(PE_j)}{\sum_{j=0}^{P-1} f(PE_j)} \tag{3}$$

$f(PE_j)$ represents the frequency of the processor PE_j and and P represents the number of PEs.

Communication This cost function gives the total amount of communication between all the PEs.

$$Commcost = \sum_{e_i \in E} volume(e_i) \times Distance[PE(Src(e_i)), PE(Snk(e_i))] \quad (4)$$

where E is the set of edges in the application task graph, $volume(e_i)$ is the amount of data exchanged by the tasks connected by the edge e_i. *Src(e_i)* and *Snk (e_i)* represent respectively the source and the sink tasks of the edge e_i. $PE(T_i)$ gives the PE on which the task T_i is mapped. *Distance(PE_1, PE_2)* gives the distance (the hop count number) between PE_1 and PE_2.

Energy consumption This cost function estimates the total energy consumption of the system under consideration as follows:

$$E_{total} = E_p + E_{comm} \quad (5)$$

where E_p is the processing energy and E_{comm} is the communication energy. Let E_{ij} be the energy needed to execute the task T_i on a processor PE_j. The processing energy can be computed as follows:

$$Ep = \sum_{i=0}^{NBT-1} \sum_{j=0}^{P-1} x_{ij} \times E_{ij} \quad (6)$$

where P represents the number of processors (PEs) and NBT represents number of tasks. The communication energy is estimated with the same model as the one given in [23]:

$$E_{bit} = E_{S_{bit}} + E_{L_{bit}} \quad (7)$$

where $E_{S_{bit}}$and $E_{L_{bit}}$ represent respectively the energy consumed on the switch and on the output link of the router. By using the preceding equation, the average energy consumption for sending one bit of data from PE_i to PE_j can be computed as follows:

$$E_{bit}^{i,j} = (nhops + 1) \times E_{S_{bit}} + nhops \times E_{L_{bit}} + 2 \times E_{local} \quad (8)$$

where *nhops* is the hop count number between routers and E_{local} represents the energy consumed by the link between the router and PEs. We can determine the total communication energy as follows:

$$E_{comm} = \sum_{e_i \in E} volume(e_i) \times E_{bit}^{PE(src(e_i)),PE(snk(e_i))} \quad (9)$$

where *src(e_i)* and *snk(e_i)* represent respectively the source and sink tasks of the edge e_i and PE(T_i) gives the PE on which the task T_i is mapped. $volume(e_i)$ is the amount of data exchanged by the tasks connected by the edge e_i. Note that this model does not consider the energy consumed by buffers in presence of contentions. For this purpose, the designer can specify the simulation model presented below (Section 4.3).

Overall completion time (Schedule Length) To define the total execution time of an application, we have defined two attributes $ST(T_i, PE_j)$ and $FT(T_i, PE_j)$ which represent respectively the Start Time and the Finish Time of the task T_i on the processor PE_j. The values of ST and FT are calculated as follows :

$$ST(T_i, PE_j) = \begin{cases} max(0, FT(T_j, PE_j)), \text{if } pred(T_i) = 0 \\ \\ max(FT(T_j, PE_j), \max_{T_k \in pred(T_i)}(FT(T_k, PE_k) + Tcomm_{ki})), \text{else} \end{cases}$$

$$FT(T_i, PE_j) = ST(T_i, PE_j) + C_{ij} \tag{10}$$

where $pred(T_i)$ is the set of the predecessors of T_i, $FT(T_j, PE_j)$ represents the end time of the last task executed on the same processor PE_j where T_i is mapped. $FT(T_k, PE_k)$ is the end time of the task T_k, where $T_k \in pred(T_i)$. $Tcomm_{ki}$ is the communication time between two tasks T_k and T_i. Equation (10) gives the end time of the execution of the task T_i. C_{ij} represents the execution time of the task T_i on the processor PE_j. Once all tasks are mapped, the total execution time of the application T_{total} is given by equation (11) as follows:

$$T_{total} = \max_{T_i \in V} FT(T_i) \tag{11}$$

As for energy model, this model does not take contentions into account.

Simulation model In addition to the analytical model, to compute cost functions, a discrete event-based simulation model has been developed. The advantage of this model over analytical one is that it takes into account the waiting time caused by simultaneous accesses to shared resources (e.g. router output ports). As previously described in [24], the proposed model is composed of an event list ($List_{event}$) to store the system events in a chronological order. The simulation consists of extracting and processing the events one by one until the event list becomes empty (i.e. application's tasks are completed) (see Algorithm 2). Each event occurs at a particular instant of time and generates other events that will be inserted into the event list using Schedule() method. Note that the simulation clock ($current_{Time}$) is advanced to the time of the next event. The *peek*() method retrieves the first item in the list, the retrieved item is not deleted and the *getWhen*() method gives the time when the event occurred. In this work, according to the assumed architecture model, the following events have been considered:

-Event 1. Execute_Task (T_i, PE_j): the simulation starts by executing this first event where the ready tasks of each processor ($R_{list}(PE_j)$) can start their execution if the processor to which they are assigned is free as given by Algorithm 1. $FT(T_k, PE_j)$ is the finish time of the last task running on the same processor PE_j where T_i is mapped and *delta* is the waiting time required to release the processor PE_j so that the task T_i can start its execution on it.

-Event 2. Generate_Packets: once a given task with successors completes its execution, this second event occurs. It consists of generating packets for each

Algorithm 1 $Execute\_Task(T_i, PE_j)$

if $(PE_j.state = free)$ **then**
  $ST(T_i, PE_j) = currentTime$
  $T_i.state = ASSIGNED$
  $PE_j.state = busy$
  $FT(T_i, PE_j) = ST(T_i, PE_j) + C_{ij}$
  $Schedule(Generate\_Packets, FT(T_i, PE_j))$
  $R_{list}(PE_j).remove(T_i)$
else
  delta= $FT(T_k, PE_j) - currentTime$
  $Schedule(Execute\_Task(T_i, PE_j), currentTime$+delta)
end if

communication. These packets are stored at the network interface's buffer. Each packet contains a set of flits mainly: the header, the payload and the tail.

-Event 3. Transfer_Flits (PE To Router): packet's flits will be sent flit by flit from the processor to the router according to the flow control technique assumed.

-Event 4. Flit_ArrivesAtInputBuffer (Router): after crossing the link between the source processor and router, the flit arrives at input buffer of this router.

-Event 5. Apply_Routing: as soon as a header flit arrives at the router's input buffer, the next hop is calculated according to the assumed routing protocol.

-Event 6. Apply_Arbitration: this event occurs when several packets request the same output port. In this case, the arbitration policy is applied to select the winner packet.

-Event 7. Traverse_Router: the winner packet sends its flits one by one through the router (crossbar) if there is enough space in its output buffer.

-Event 8. Flit_ArrivesAtOutputBuffer (Router): after crossing the router's crossbar, flit arrives at the router's output buffer. If the entire packet has arrived (i.e. flit is queue), a new arbitration for this output buffer can start.

-Event 9. Traverse_Link (Router-Router): according to the flow control technique considered, flits are transmitted between two neighboring routers.

-Event 10. Transfer Flits (Router-Processor): this event occurs if the packet's final destination corresponds to the router's local port.

-Event 11. Flit_ArrivesAtInputBuffer (NI): at the end, the flit arrives at the network interface of the destination processor, where a phase of packet arrival control (flits) and initial message formation will take place.

Overall Completion Time using Simulation Model: the total execution time of a given application corresponds to the time elapsed between the execution of the first event and the execution of the last event given by T_{total} (see the Algorithm 2).

Energy consumption using Simulation Model: unlike the analytical model given above, this model takes into account routing, arbitration and buffer's energies when computing communication energy E_{comm}. It should be noted that

Algorithm 2 proposed simulation model

$List_{event}$: is the list of events considered
NB_{TT}: is the number of completed tasks initialized to 0
NB_{TTotal}: is the total number of tasks in a given application
$current_{Time}$: is the current time initialized to 0

```
Determine the R_list(PE_j) (See Algorithm 3)
for each processor PE_j do
  if R_list(PE_j) is not empty then
    T_i = R_list(PE_j).peek()
    Schedule(Execute_Task(T_i, PE_j), currentTime)
  end if
end for
while (NB_TT < NB_TTotal) do
  while (List_event is not empty) do
    event = List_event.peek()
    if (event.getWhen() > currentTime) then
      next_time = event.getWhen()
      break
    end if
    event.run()
    List_event.remove(event)
  end while
  currentTime = next_time
end while
T_total = currentTime
```

Algorithm 3 Determine the $R_{list}(PE_j)$

T_{list} : is a list containing the application's tasks ordered with a given scheduling policy

```
for each task T_i ∈ T_list do
  T_i.state = INIT
  if (nb(T_i.pred()) = 0 and x_{i,j} = 1) then
    R_list(PE_j).add(T_i)
    T_i.state = READY
  end if
end for
```

this value increases proportionally with the waiting time generated during contentions.

4.4 Application and architecture constraints

The proposed tool offers a very effcient mechanism to associate a set of constraints with each solution. The following constraints can be specified by the designer:

Task assignment Each task is assigned to exactly one processor, i.e:

$$\sum_{j=0}^{P-1} x_{ij} = 1, \forall i \in [0, NBT - 1] \tag{12}$$

where P is the number of processors (PEs) and NBT is the number of tasks.

Deadline constraint We can set a deadline for each task or application. In these two cases their finish time should be less than their deadline.

Pre-assignment In some cases, one can predefine a tasks assignment on specific processors (like dedicated accelerators) for better performance purposes.

4.5 Mapping problem

As mentioned above, the mapping problem involves optimization of several objectives (often conflicting) simultaneously. So our goal is to find the Pareto optimal mapping solutions set. For this purpose, in addition to multiobjective optimization algorithms included in jMetal framework [25] which we have adapted to solve mapping problem in our previous work [9], we have added the following multiobjective algorithms:

-AMOSA [10]: is a multiobjective vesion of Simulated Annealing (SA). Its principal features are: (1) It integrates an archive to store non-dominated solutions found during the search. (2) The archived solutions are also used to determine the probability of acceptance of a new solution (active elitism). The size of the archive is kept limited. (3) AMOSA uses the concept of amount of domination in the calculation of the probability of acceptance of a degrading solution.

-PMOTS [11]: is a multiobjective version of Tabu Search TS, called PMOTS, which means "Parallel-MultiObjective Tabu Search". The algorithm exploits K parallel search trajectories. A tabu list is assigned to each search path. This list is used to prevent cycles and to force the acceptance of degrading solutions (dominated solutions), in order to direct the search to new regions not yet visited. The non-dominated solutions found during the search will be saved in a list. This list will contain the final non-dominated solutions (the optimal Pareto front).

-MBB [12]: is a multiobjective version of Branch & Bound based on Pareto's dominance. The search space is explored by dynamically building a tree whose root node represents the problem being solved and its whole associated search space. The leaf nodes are the potential solutions and the internal nodes are subproblems of the total solution space. The size of the subproblems is increasingly reduced as one approaches the leaves. The construction of such a tree and its exploration are performed using two main operators: branching and pruning [22].

-Hybrid metaheuristics: we have also combined metaheuristics which gives new ones called hybrid metaheuristics. For example, instead of initializing the AMOSA's archive or PMOTS's parallel search randomly, we have used the solutions returned by population-based metaheuristics offered by our tool.

Solving Mapping problem using AMOSA [10] and PMOTS [11] To apply these algorithms to the mapping problem, a solution representation and a corresponding neighborhood move operator are to be specified.

- Solution Representation: the potential solution (point) in AMOSA and PMOTS algorithm is like the chromosome representation given in [9].

- Neighborhood Move Operator: all mutations' type provided by jMetal framework [25] can be specified as neighborhood move operator for AMOSA and PMOTS algorithms. These operators have been adapted to solve the mapping problem in [9].

Solving Mapping problem using MBB [12] In this paper, we have used an exact method to check the efficiency of the metaheuristics presented in the tool in solving small and medium mapping problem instances. For this, we have defined these two main operators:

- The branching strategy: it determines the order in which the branches are explored [22]. In our case, at each search level, the non-dominated solutions are explored first (i.e. the best-first strategy), and if more than one non-dominated solutions are found, the depth-first strategy is applied.

- The prunning strategy: it eliminates the partial solutions that do not lead to optimal solutions by computing the lower bound associated with a partial solution [22]. In our work, at each level, the lower bound corresponds to the best solution's evaluation which can be found starting from the partial solution of this level. If the lower bound of a node (partial solution) is dominated by a given solution in the upper bound, the exploration of the node never leads to optimal solution, so the node is ignored. Note that, upper bound set corresponds to a set of non-dominated solutions generated randomly. Figure 3 presents an illustration of the MBB [12] method on the mapping of three tasks (T_0, T_1, T_2) and each task has two permissible processors (PT_0={0, 1}, PT_1={2, 3} and PT_2={6, 7}).

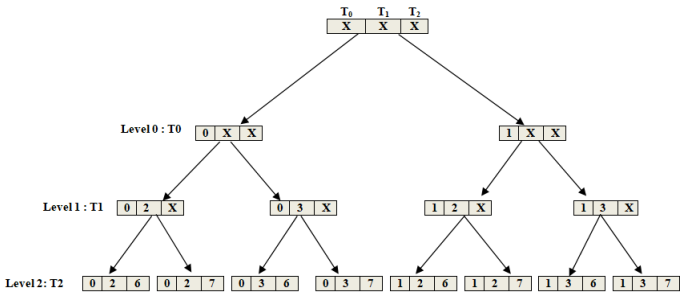

Fig. 3. Illustration of the MBB [12] method on the mapping problem

5 EXPERIMENTAL RESULTS

In this section, we present a set of experiments illustrating the use of our proposed mapping tool to solve several mapping problems' instances (small, medium and large). These instances differ from each other regarding the task graphs and platform size used. TGFF [26] is used to generate a set of task graphs randomly. The architecture model (Platform) consists of k types of processors interconnected using NoC topology. Table 1 gives the NoC parameters used in the following experiments. To evaluate and compare multiobjective optimization

Table 1. NoC's parameters

NoC topologies	2D-Mesh, 2D-Torus and Spidergon
Switching technique	Wormhole switching
Routing technique	XY routing algorithm (2D-Mesh and 2D-Torus)
Arbitration technique	Round Robin (RR)
Flow control	Credit-based

algorithms performances, two properties are usually required: convergence and a uniform diversity. Several quality indicators for measuring these two criteria are included in jMetal framework [25]. In this paper, we have considered the following evaluation metrics: Inverted Generational Distance (IGD)(convergence and diversity measure)([27] and Epsilon (convergence measure) [25]. The smaller the IGD and Epsilon values are, the better the quality of the obtained solutions. Since some of the applied metrics require an optimal Pareto set to be computed, we have constructed it by collecting the results of all the runs of the different algorithms. In all our experiments, we have performed 30 independent runs and the obtained tables represent the mean and standard deviation values of the quality indicator applied. The best and the second best metaheuristics are marked with different levels of gray as background color. TABLE 2 gives algorithm's parameter setting used in the following experiments. A PC Intel (R)

Core (TM) i7 CPU, 2.7GHz, with 8 GB of RAM is used to perform all the calculations.

Comparative study of the different algorithms

To make a comparison between the algorithms offered by our mapping tool, we have considered several cases:

- Case1 - *Variation of task graphs and platforms used*- Figure 4 gives comparaison between several multiobjective optimization algorithms offered by our mapping tool (NSGAII, FastPGA, SPEA2, PESAII, IBEA, SMPSO, PAES, FastPGA, OMOPSO, AbYSS, MOCell, AMOSA, PMOTS and MBB) for solving small (P1), medium (P2) and large (P5) mapping instances (see Figure 4 (a, b and c) respectively). We have fixed some tasks' processors assignement (i.e. pre-assignement constraint). Two conflicting cost functions are optimized: the energy consumption and the overall completion time defined using analytical model and Epsilon and IGD are the quality indicators used to assess the performance of the fronts returned by the different algorithms. These experiments show that the performance order of multiobjective algorithms changes according to the mapping problem considered (task graph and platforms used). For example, for small and medium mapping problems, the two best algorithms are respectively MBB and PMOTS, while for large mapping instance, the best results are given by SMPSO and OMOPSO respectively. This confirms that a mapping tool must provide several metaheuristics in order to explore different solution spaces. We can also see that the algorithms' execution time increases with the size of the problem (task graph and platform used). An example of some algorithms' runtime for solving small, medium and large mapping problems is given in TABLE 4.

- Case2 - *Variation in the number of objectives to be optimized (NObj)*- The experiments presented in this second case give a comparison between a set of algorithms offered by our mapping tool (NSGAII, SPEA2, MOCell, AbYSS, SMPSO, OMOPSO, AMOSA and PMOTS) by specifying two objectives (load balancing, communication cost), three objectives (load balancing, communication cost and energy consumption) and four objectives (load balancing, communication cost, energy consumption and overall execution time) (see Table 5). These cost functions are defined by analytical models. From Table 5, we see that, firstly, for the same mapping problem (the problem P3 in our case), the performance of metaheuristics changes according to the number of objectives specified. For instance, in these experiments, the results returned by OMOPSO algorithm are poor when considering two objectives, but they are good in the case of three and four objectives. Secondly, these experiments show that metaheuristics' execution time increases with the increase in the number of objectives considered. For example, the execution time of NSGAII for 2, 3 and 4 objectives is respectively $0.560s$, $0.950s$ and $6.791s$.

- Case3 - *Effect of the choice of metaheuristic parameters on the quality of the returned mapping solutions*- In this third case, we present a set of other experiments showing how metaheuristics are sensitive to their parameters. Figure 5

Table 2. Algorithms' Parameterization

Population-based metaheuristics	(NSGAII, FastPGA, SPEA2, PESAII, IBEA, MOCell, AbYSS, SMPSO and OMOPSO)
NSGAII/ FastPGA	
Population Size	100
Max Iterations	25000 / 10000 (case 4)
Mutation Probability	1.0/L (L : individual length)
Crossover Probability	0.8 / 0.9 (case 4)
SPEA2/ PESAII/ IBEA	
Population Size	100
Archive Size	100
Max Iterations	25000 / 10000 (case 4)
Mutation Probability	1.0/L (L : individual length)
Crossover Probability	0.8 / 0.9 (case 4)
MOCell	
Population Size	100
Archive Size	100
Max Iterations	25000
feedback	20
Mutation Probability	1.0/L (L : individual length)
Crossover Probability	0.8
AbYSS	
Population Size	100
Archive Size	100
Max Iterations	25000
Reference Set Size	10 + 10
Mutation Probability	1.0/L (L : individual length)
Crossover Probability	1.0
SMPSO/ OMOPSO	
Swarm Size	100
Max Iterations	500
Archive Size	100
Mutation Probability	1.0/L (L : individual length)
Single-based metaheuristics	(AMOSA, PMOTS and PAES)
AMOSA	
Initial temperature (T_0)	700 / 800 (case 4)
Final temperature (T_1)	10^{-3}
Cooling rate (α)	0.8 / 0.9 (case 4)
Max Iterations	1000
Hard Limit (HL)	100
Soft Limit (SL)	110
Gamma	1.7 / 1.8 (case 4)
PMOTS	
Max Iterations	5000
Max Pareto Rank (R_{max})	4
Max Neighbours	50
Number Parallel search paths (K)	10
Size min of Tabu List (Tab_{min})	10
Size max of Tabu List (Tab_{max})	15
PAES	
Archive Size	100
Max Iterations	25000
Mutation Probability	1.0/L (L : individual length)

Table 3. mapping problems' instances (MPs)

MP	Task graph	Platforms
P1	7 tasks	3x3 2D-mesh with 3 processors' types
P2	10 tasks	4x4 2D-mesh with 3 processors' types
P3	100 tasks	8 processors interconnected with Spidergon topology (3 processors' types)
P4	100 tasks	4x4 2D-mesh with 3 processors' types
P5	100 tasks	8x8 2D-torus with 6 processors' types

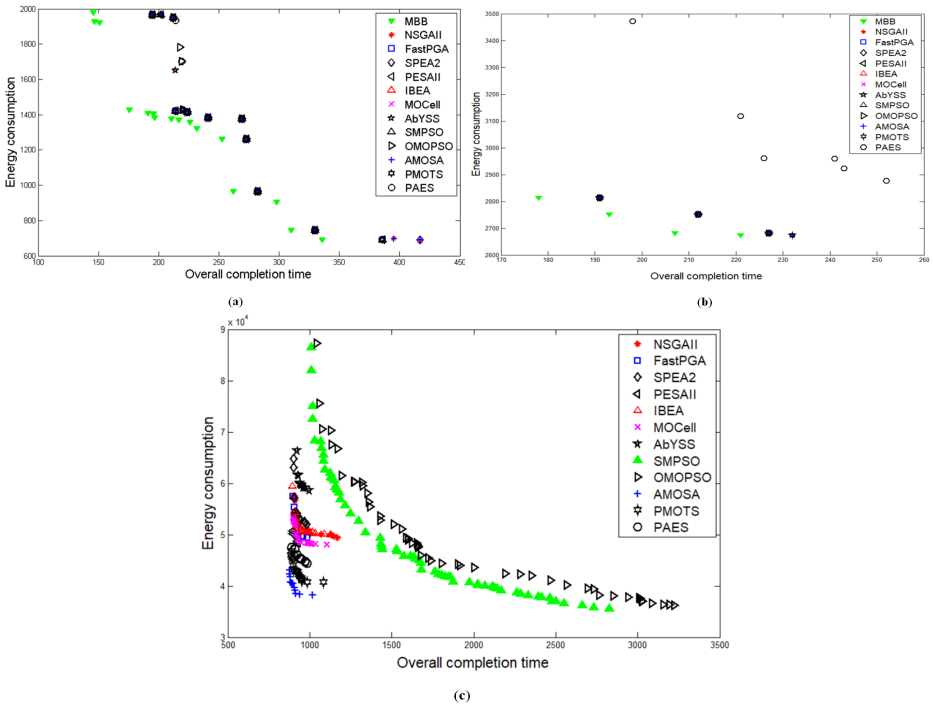

Fig. 4. Algorithms' comparison: (a) Solving small mapping instance, (b) Solving medium mapping instance, (c) Solving large mapping instance.

MPs	NSGAII	IBEA	MOCell	AbYSS	SMPSO	OMOPSO	AMOSA	PMOTS	PAES	MBB
P1 (small)	0.235s	2.027s	0.177s	0.169s	0.222s	0.216s	0.242s	11.653s	0.122s	3.595m
P2 (medium)	0.272s	2.080s	0.209s	0.208s	0.302s	0.287s	0.384s	11.720s	0.153s	17.95m
P5 (large)	3.560s	5.496s	3.572s	3.499s	6.166s	6.064s	7.577s	5.601m	3.0576s	–

Table 4. Algorithms' runtime

shows several experiments on the effect of some parameters like: the evaluations' number ($MaxEval$), an example of AMOSA's parameter (α), and mutation operator on the performance of some metaheuristics proposed by our mapping tool such as: SMPSO, AMOSA and SPEA2. In all these experiments, we varied one parameter for each algorithm and other parameters were set. The study was done on P3 problem presented in Table 3. Load balancing and communication were specified as objective functions to be optimized in these experiments. Figure 5(a) shows clearly that the more the evaluations are, the better the quality of the mapping solutions found at the expense of the execution time (see Table 1 in Figure 5). Another example of a parameter affecting the AMOSA algorithm is given in Figure 5(b). It is the parameter α whose value varies between $]0, 1[$. The closer the value of this parameter is to 1, the better is the performance of AMOSA, also at the expense of its execution time (see Table 2 in Figure 5).

NObj			NSGAII	SPEA2	MOCell	AbYSS	SMPSO	OMOPSO	AMOSA	PMOTS
2	IGD	MOY	$8.18e-02$	$8.86e-02$	$8.13e-02$	$8.30e-02$	$2.78e-02$	$1.16e-01$	$7.88e-02$	$8.99e-02$
		ET	$8.5e-03$	$8.5e-03$	$6.8e-03$	$6.9e-03$	$1.4e-03$	$1.5e-02$	$5.9e-03$	$3.2e-03$
		Runtime	0.560s	1.144s	0.568s	0.368 s	1.086s	0.777s	0.349s	2.522s
3	IGD	MOY	$6.59e-02$	$6.91e-02$	$6.56e-02$	$5.33e-02$	$2.72e-03$	$1.82e-02$	$5.72e-02$	$6.50e-02$
		ET	$5.6e-03$	$4.8e-03$	$3.7e-03$	$8.3e-03$	$1.5e-04$	$6.0e-04$	$2.1e-03$	$2.0e-03$
		Runtime	0.950s	1.580s	0.928s	0.727s	1.527s	1.325s	0.930	5.139s
4	IGD	MOY	$2.17e-02$	$2.41e-02$	$2.20e-02$	$1.61e-02$	$6.50e-03$	$1.21e-02$	$2.53e-02$	$2.44e-02$
		ET	$1.5e-03$	$1.1e-03$	$1.1e-03$	$2.7e-03$	$3.9e-04$	$4.4e-04$	$5.0e-04$	$4.6e-04$
		Runtime	6.791s	7.162s	6.687s	8.910s	265.742s	377.501s	8.953s	35.616s

Table 5. IGD. Mean and standard deviation

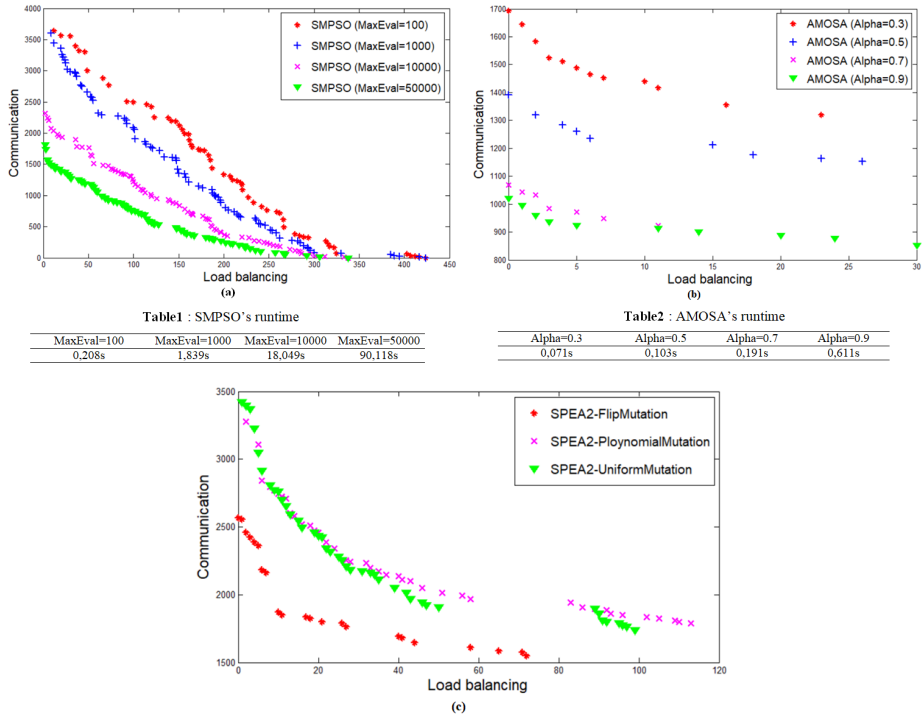

Table1 : SMPSO's runtime

MaxEval=100	MaxEval=1000	MaxEval=10000	MaxEval=50000
0,208s	1,839s	18,049s	90,118s

Table2 : AMOSA's runtime

Alpha=0.3	Alpha=0.5	Alpha=0.7	Alpha=0.9
0,071s	0,103s	0,191s	0,611s

Fig. 5. Algorithms' parameters setting effect: (a) varying of the evaluations' number (MaxEval) of SMPSO algorithm, (b) varying AMOSA's cooling rate (α), (c) varying SPEA2's mutation operators.

Other important parameters affecting the performance of metaheuristics are the reproduction operators: (1) mutation (type and probability) and (2) crossover (type and probability). This is clearly demonstrated in the experiment shown in Figure 5 (c) which indicates that *Flip mutation* operator gives better results compared to the other types of mutation operators (polynomial and uniform mutations).

- Case4 -*Comparing between hybrid and non hybrid metaheuristics*- As the last experiments, we compared hybrid (HNSGAII, HSPEA2 and HPESAII) and non hybrid metaheuristics (NSGAII, SPEA2 and PESAII) for solving P5 problem (see Table 3). Two conflicting cost functions are optimized: the energy consumption and the overall completion time defined using simulation model. From Figure 6, we see that the proposed hybrid algorithms give promising results (better fronts) compared to non-hybrid algorithms at the expense of time (see Table 6). This extra time of the hybrid algorithms compared to the non-hybrid algorithms in these experiments is due to the AMOSA runtime used for the improvement of the Pareto fronts resulting by NSGAII, SPEA2 and PESAII metaheuristics. It is important to note that the parameters of the hybid algorithms are a combination of those given in Table 2 (with the same configuration values).

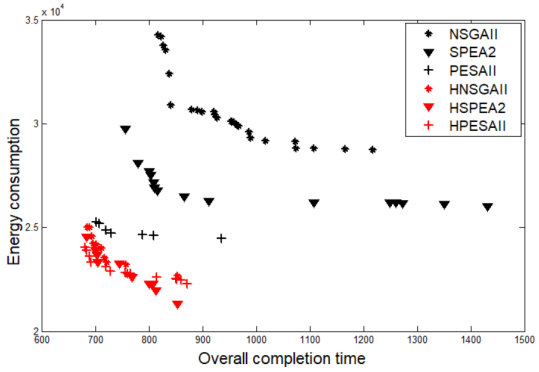

Fig. 6. comparative study of hybrid and non hybrid algorithms.

	NSGAII	HNSGAII	SPEA2	HSPEA2	PESAII	HPESAII
Runtime (in minutes)	3.12m	6.354m	2.933m	6.121m	2.639m	5.83m

Table 6. Runtime of hybrid and non hybrid algorithms

5.1 Discussion

The proposed mapping tool offers several multiobjective algorithms that can be specified when solving the mapping problem. The question is: "What is the most efficient algorithm (quality of solutions and runtime) to solve a given mapping problem?" In this section, we try to summarize our experimental study, and thus we give a series of guidelines on the use of the proposed tool. According to the experimental results presented above, we concluded that:

- For small (or medium) instances of the mapping problem, the exact method (MBB) as well as metaheuristics with "good parameter" give good compromise solutions (i.e. good Pareto fronts). The exact method (MBB) gives the exact Pareto front. Therefore, the exact method (MBB) is preferable in this case (see case 1).

- For large mapping problems' instances, only metaheuristics can be used. The exact method (MBB) does not give results in polynomial time considering a large search space. Therefore, the exact method (MBB) cannot solve large mapping problems' instances (see Table 4 in case 1).

- Although metaheuristics can deal with all mapping problems' instances (small, medium or large), their main disadvantage is their sensitivity to parameters. Therefore, a sensitivity analysis must be done in order to estimate the right parameters for each algorithm. It should be noted that there is no standard optimal setting for metaheuristics since this is strongly related to the mapping problem under consideration (see case 3).

- No metaheuristic is better than another for any kind of mapping problem. This depends on several factors such as: the task graph and the platform used (see case 1), objectives' number specified (see case 2) and metaheuristics' parameters (see case 3).

- Metaheuristics' execution time (runtime) depends on several factors such as: the problem's size (task graph and platforms used) (see Table 4 in case 1), the number of objectives to be optimized (case 2), the parameters' configuration of the algorithm (see Table1 and Table2 in Figure 5) and the type of objective function specified (analytical (case 1) vs. simulation (case 4)).

- The solution-based metaheuristics, for example AMOSA can effectively improve P-metaheuristics like NSGAII, SPEA2 and PESAII (see Figure 6). The major disadvantages of these resulting hybrid metaheuristics are their additional runtime compared to non hybrid ones (see case 4) and the difficulty of configuring their parameters, since they consist of a combination of parameters of the metaheuristics which are, also by their nature, very sensitive to their parameters.

6 CONCLUSION

In this paper, a new tool for mapping applications on NoC based on heterogeneous MPSoCs is proposed, in which several multiobjective optimization algorithms can be specified to explore the mapping space. Our tool offers the designer the flexibility to easily add a new cost function or any application and architecture constraints. It offers also an easy way to assess the performance of the front returned by different algorithms. In our future work, we are planning to consider real time constraints during the mapping and generalize the mapping solution proposed in this paper in simultaneous mapping of several applications (critical and/or non-critical) on the same target platform.

References

1. U. U. Tariq, H. Wu, and S. A. Ishak: Energy and memory-aware software pipelining streaming applications on NoC-based MPSoCs. Future Generation Computer Systems**111**, 1–16 (2020)
2. H. Javaid and S. Parameswaran: Pipelined Multiprocessor System-on-Chip for Multimedia. Springer, Cham (December 2013)
3. L. Suriano, A. Otero, A. Rodríguez, M. Sánchez-Renedo, and E. De La Torre: Exploiting Multi- Level Parallelism for Run-Time Adaptive Inverse Kinematics on Heterogeneous MPSoCs. IEEE Access **8**, 118707–118724 (2020)
4. STMicroelectronics, Nomadik application processor, http://www.st.com
5. NVIDIA Tegra: Next Generation Mobile Development, https://developer.nvidia.com/tegra-development
6. Mobile processor exynos 5 octa (5422), https://www.samsung.com/semiconductor/minisite/exynos/products/mobileprocessor/exynos-5-octa-5422/.
7. L. Benini, G. De Micheli: Networks on Chips: A new SOC paradigm. IEEE Computer **35**(1), 70–78 (2002)
8. Michael R. Garey and David S. Johnson.: Computers and Intractability: A Guide to the Theory of Np-Completeness. W.H.Freeman & Co Ltd, New York (1979)
9. D. Belkacemi, Y. Bouchebaba, M. Daoui, and M. Lalam.: Network on Chip and Parallel Computing in Embedded Systems. In: 2016 IEEE 10th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSOC), pp. 146–152. IEEE, (September 2016)
10. S. Bandyopadhyay, S. Saha, U. Maulik, and K. Deb.: A Simulated Annealing-Based Multiobjective Optimization Algorithm: AMOSA. IEEE Transactions on Evolutionary Computation **12**(3), 269–283 (June 2008)
11. K. Jaffrs-Runser, J-M. Gorce, and C. Comaniciu: A Multiobjective Tabu Framework for the Optimization and Evaluation of Wireless Systems. In: Wassim Jaziri, Editor, Tabu Search. I-Tech Education and Publishing, Vienna, Austria (2008). https://doi.org/ ISBN 978-3-902613-34-9
12. J. Carlos Soto-Monterrubio, Alejandro Santiago, H. J. Fraire- Huacuja, Juan Frausto-Solís, and J. David Terán-Villanueva: Branch and Bound Algorithm for the Heterogeneous Computing Scheduling Multi-Objective Problem. International Journal of Combinatorial Optimization Problems and Informatics **7**(3), 7–19 (2016)
13. G. Ascia, V. Catania, and M. Palesi: Mapping Cores on Network-on-Chip. International Journal of Computational Intelligence Research **1**(2), 109–126 (2005)
14. C. Erbas, S. Cerav-Erbas, and A.D. Pimentel: Multiobjective optimization and evolutionary algorithms for the application mapping problem in multiprocessor system-on-chip design. IEEE Transactions on Evolutionary Computation **10**(3), 358–374 (June 2006)
15. W. Zhou, Y. Zhang and Z. Mao: Pareto based Multi-objective Mapping IP Cores onto NoC Architectures. Circuits and Systems, APCCAS (2006)
16. R. Tornero, V. Sterrantino, M. Palesi, and J. M. Orduna: A multi-objective strategy for concurrent mapping and routing in networks on chip. 2009 IEEE International Symposium on Parallel Distributed Processing, pp. 1–8. IEEE, (May 2009)
17. N. Nedjah, M. Vinícius Carvalho da Silva, and L. de Macedo Mourelle: Customized computer-aided application mapping on NoC infrastructure using multi-objective optimization. Journal of Systems Architecture **57**(1), 79–94 (January 2011)
18. N. Wu, Y. Mu, and F. Ge : GA-MMAS: an Energyand Latency-aware Mapping Algorithm for 2D Network-on-Chip. IAENG International Journal of Computer Science **39**(1), (2012)

19. T. He and Y. Guo: Power consumption optimization and delay based on ant colony algorithm in network-on-chip. Engineering Review **33**(3), 219–225 (2013)
20. N. Chatterjee, S. Reddy, S. Reddy, and S. Chattopadhyay: A reliability aware application mapping onto mesh based Network-on-Chip. In 2016 3rd International Conference on Recent Advances in Information Technology (RAIT), pp. 537–542. (March 2016)
21. J.V. Bruch, E.A. da Silva, C.A. Zeferino, and L.S. Indrusia: Deadline, Energy and Buffer-Aware Task Mapping Optimization in NoC-Based SoCs Using Genetic Algorithms. In 2017 VII Brazilian Symposium on Computing Systems Engineering (SBESC), pp. 86–93. IEEE, (November 2017)
22. El-Ghazali Talbi: Metaheuristics: from design to implementation. John Wiley & Sons, Hoboken, N.J, New Jersey and canada (2009)
23. Jingcao Hu and R. Marculescu: Energy-aware mapping for tilebased NoC architectures under performance constraints. In Proceedings of the ASP-DAC Asia and South Pacific Design Automation Conference, pp. 233–239. IEEE, (January 2003)
24. D.Belkacemi, : Parallel Applications Mapping onto Network on Chip Based on Heterogeneous MPSoCs Using Hybrid Algorithms. International Journal of Distributed Systems and Technologies **10**(2), (2019)
25. Juan J. Durillo and Antonio J. Nebro: jMetal: A Java framework for multi-objective optimization. Advances in Engineering Software **42**(10), 760–771, (October 2011)
26. R.P. Dick, D.L. Rhodes and W. Wolf: TGFF: task graphs for free. Workshop on Hardware/Software Codesign, (1998)
27. Antonio J. Nebro, Francisco Luna, Enrique Alba, BernabÉ Dorronsoro, Juan J. Durillo, and Andreas Beham: AbYSS: Adapting Scatter Search to Multiobjective Optimization. IEEE Transactions on Evolutionary Computation **12**(4), 439–457 (August 2008)