Journal of Systems Architecture 59 (2013) 429–440

Contents lists available at SciVerse ScienceDirect

Journal of Systems Architecture

journal homepage: www.elsevier.com/locate/sysarc

From UML specifications to mapping and scheduling of tasks into a NoC, with reliability considerations

F. Bolanos [a,*], F. Rivera [b], J.E. Aedo [b], N. Bagherzadeh [c]

[a] National University of Colombia, Medellin, Calle 59A Num. 63-20, Medellin, Colombia
[b] University of Antioquia, Calle 70 Num. 52-21, Medellin, Colombia
[c] The Henry Samueli School of Engineering, University of California, Irvine, CA 92697-2625, United States

ARTICLE INFO

Article history:
Received 16 August 2012
Received in revised form 18 February 2013
Accepted 30 April 2013
Available online 15 May 2013

Keywords:
Network-on-Chip (NoC)
Unified Modeling Language (UML)
Fault tolerance
Application mapping
Multiprocessor System-on-Chip (MPSoC)

ABSTRACT

This paper describes a technique for performing mapping and scheduling of tasks belonging to an executable application into a NoC-based MPSoC, starting from its UML specification. A toolchain is used in order to transform the high-level UML specification into a middle-level representation, which takes the form of an annotated task graph. Such an input task graph is used by an optimization engine for the sake of carrying out the design space exploration. The optimization engine relies on a Population-based Incremental Learning (PBIL) algorithm for performing mapping and scheduling of tasks into the NoC. The PBIL algorithm is also proposed for dynamic mapping of tasks in order to deal with failure events at runtime. Simulation results are promising and exhibit a good performance of the proposed solution when problem size is increased.

1. Introduction

Complexity of embedded systems and their applications has grown continuously in later years. As a consequence of such a trend, developers are now compelled to conceive optimal ways in order to perform the design process. Nowadays, platforms are composed of several heterogeneous processing elements (PEs) which are capable of executing applications in a concurrent fashion. Multiprocessor System-on-Chip (MPSoC) is a current paradigm for dealing with the increasing demand for performance, as well as constraints such as power consumption or real time. In order to deal with the ever increasing amount of PEs in the system, interconnection among cores or processors becomes a key issue. An ideal interconnection system must provide full connectivity among the set of PEs which compose the system, minimize the communications delay between any pair of PEs, use the interconnection resources efficiently, and scale easily as the number of PEs increases. Network-on-Chip (NoC) is an interconnection paradigm widely used which represents the best current tradeoff among these issues. Apart from the constraints already mentioned, designer also faces a non-trivial issue of distributing tasks efficiently among the available resources in the system.

System Level Design (SLD) is a model-based approach aimed to increase the throughput in the design process [1]. The key in SLD is the modeling of the platform and applications features in an independent way in the early stages of the design process. Models allow the early specification of the application, and the estimation of the platform properties before the implementation of the system. Then, the design space exploration may take place, allowing the making of key implementation decisions very early in the design process.

In SLD, the design process involves the use of two kinds of models: the Platform Independent Model (PIM), and the Platform Specific Model (PSM). PIM models are used as an initial specification of the system, without taking into account implementation details. A PIM describes the behavior and constraints of the system at hand. PSM is a link between the PIM and a specific implementation platform. A PSM provides information about the behavior of the application running on a given platform, as well as allows inferring some figures of merit such as execution time, power consumption, and resources usage.

In developing of the PSM, static mapping and scheduling are two key stages, which consist in deciding the physical placement in the NoC of each processing element, as well as its workload, in terms of executable tasks. This paper proposes a methodology for performing mapping and scheduling of executable tasks starting from a PIM specification, which takes the form of a UML model [2].

The starting point of such a proposal is an UML specification of the intended application, which consists of two diagrams, i.e., Class and State Machine. A transformation tool chain is responsible for taking the UML model and generating an intermediate representation, which highlights the data dependences, implementation

* Corresponding author. Tel.: +57 4 5703257.
E-mail addresses: fbolanosm@unal.edu.co (F. Bolanos), farivera@udea.edu.co (F. Rivera), joseaedo@udea.edu.co (J.E. Aedo), nader@uci.edu (N. Bagherzadeh).

1383-7621/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.sysarc.2013.04.009

figures of merit, and design constraints. This intermediate representation takes the form of an Annotated Acyclic Directed Task Graph (ADAG) and serves as input to the optimization engine, which is founded on a Population-based Incremental Learning (PBIL) algorithm. The optimization engine performs a design space exploration and decides the workload and placement of each resource of the system.

On the other hand, this paper also proposes taking advantage of the optimization engine based on PBIL algorithms for performing dynamic mapping. Dynamic mapping is performed in execution time as a result of two potential situations: in the first place, it could be desirable reallocating the executable tasks onto the available resources as a consequence of changing operating conditions, such as network traffic or workload. Secondly, dynamic mapping is a way to provide fault tolerance. In the event of a failure of a given resource, the system may adapt itself by reallocating its associated tasks to non-faulty resources.

The proposed dynamic strategy is aimed to implement fault tolerance in the system, as a way to providing some level of reliability. If there is a failure in some of the PEs which compose the system, a PBIL algorithm may find the best reallocation scheme for the tasks which were running on the faulty resources, for the sake of guaranteeing their correct execution.

Among the main contributions of this paper it may be mentioned the availability of a holistic solution, capable of performing static mapping and scheduling of executable tasks, starting from an UML specification. The heart of such mapping and scheduling stage is an optimization engine based on a multiobjective adaptive PBIL algorithm. On the other hand, the same multiobjective adaptive PBIL algorithm is customized for the sake of performing dynamic mapping of tasks aimed to provide fault tolerance by means of the reallocation of tasks that were running in faulty processors in fault-free resources.

The remainder of this paper is organized as follows: Section 2 surveys some of the more relevant reported works in the subject of this paper. Section 3 shows and describes the proposed solution, aimed to perform mapping and scheduling, starting from the UML specification. The modifications made to the basic PBIL algorithm are presented, as well as the fault-tolerance strategy based on dynamic mapping. Section 4 shows the experimental results of testing the proposed approach in the context of a 2D NoC architecture, with a deterministic routing scheme. Final remarks and conclusions are presented in Section 5.

2. Related work

Currently, embedded systems designers face a set of conflicting elements, such as performance constraints, applications variability, reasonable prototyping times, and physical world constraints such as power consumption or real-time. SLD is an attempt to overcome these issues, and relies in the idea of using high levels of abstraction at the early stages of the design process [3,4]. Such a high level specification demands the use of a given language. There has been two main trends in this subject: Unified Modeling Language (UML) [5,6], and Architecture Analysis and Design Language (AADL) [7]. These languages are suitable for developing the PIM of the system, which may be customized and constrained by means of profiles and annotations.

Regarding the platforms, the most conspicuous trend on embedded systems design, is the integration of the whole system in a single chip (System-on-Chip, SoC). Such integration implies also the availability of several processing units (multicore and manycore systems) which are often connected by means of a Network-on-Chip (NoC).

As developing the PSM implies the modeling of the target architecture, the high level specification model must be transformed into a common domain semantic, which combines details of both the specification and the platform. Among the several options available for such a purpose, a tool called Task Graphs for Free (TGFF) offers a widely used format to generate and represent PSM models in a standard fashion by means of ADAGs [8].

The design process of an embedded system which is based on the NoC paradigm implies two critical stages: mapping and scheduling. Such stages affect a set of figures of merit which are often in conflict. For the sake of dealing with the complexity involved with mapping and scheduling, some approaches treat each problem independently [9–11]. Some other solutions [12–14] are able to perform these two stages in a joint way, but are restricted to homogeneous architectures, i.e., those multicore architectures which have a set of identical processing elements. Finally, some other reported works disregard the communication assessments in performing mapping and scheduling [15,16].

Singh et al. [17] propose a hybrid strategy for mapping and scheduling of applications into heterogeneous architectures (i.e., multicore architectures with several kinds of processing elements). The hybrid nature of such approach implies that mapping and scheduling solutions are calculated in two steps: the computation of tradeoff points and resource throughput analysis is performed in design time, because these computations are the most computing-intensive. Then, a run-time optimizer chooses the best tradeoff point for a given application which is going to run in the system. The main problem with this approach is related with its inability for finding new tradeoff points, if new applications need to be mapped onto the NoC.

The work reported in [18], is aimed to perform mapping in design time for heterogeneous architectures. Both computation and communication assessments are taken into account in the optimization process, which is performed via a simulated annealing algorithm. Such a kind of algorithm may exhibit local-optimum issues [19]. On the other hand, population-based approaches are more robust for dealing with complex optimization problems and multiobjective scenarios [20].

In the work described in [21], a solution based on genetic algorithms for performing mapping and scheduling of applications on heterogeneous architectures is described. Genetic algorithms are the most known instance of population-based techniques. The algorithm optimizes the power consumption of the final implementation, while deals with timing constraints. The target architecture is a 2-D mesh, and a wormhole routing schema has been considered for simulating the system.

Table 1 summarizes some of the reported solutions for mapping of tasks on NoC-based architectures. Several classification criteria have been taken into account. Second column of Table 1 reports whether the target architecture of the referred approach is homogeneous or heterogeneous. Third column is devoted to the taxonomy of the approaches according with the moment in which such strategies take place: static for design time, Dynamic for runtime, and hybrid. Fourth column shows the nature of the optimization engine.

Some of the reported algorithms in Table 1 are Genetic Algorithms (GA), Simulated Annealing (SA), and Integer Linear Programming (ILP). The common domain semantic or intermediate representation is depicted in fifth column of Table 1. As can be seen, task graphs are the most common solution regarding this issue. Finally, last column of Table 1 shows the objectives that are taken into account in the optimization process.

As depicted in Table 1, the current proposal targets heterogeneous architectures. In the same way, the mapping may be static or hybrid, depending on whether a recovery fault strategy is going to be implemented. The optimization engine is based on a PBIL algorithm, which is multiobjective. The intermediate representation or common domain semantic for the proposed solution corresponds to a task graph.

Table 1
Summary of the reported mapping solutions.

Reference	Target architecture	Optimization nature	Optimization algorithm	Common domain semantic	Optimization objective
Jang et al. [22]	Heterogeneous	Static	Successive relaxation or GA	Metric space	NoC traffic
Singh et al. [17]	Homogeneous	Hybrid	Custom	SDFG	Throughput
Antunes et al. [23,24]	Homogeneous	Hybrid	SA (static) and custom (Dynamic)	TCG	Energy
He et al. [25]	Heterogeneous	Static	Mixed ILP	Annotated task graph	Energy and execution time
Hosseinabady et al. [26]	Heterogeneous	Dynamic	Distributed stochastic	Task graph	Communication energy
Hamedani et al. [27]	Homogeneous	Static	ILP	Communication task graph	Peak temperature
Wang et al. [28]	Homogeneous	Dynamic	Custom	TCG	Average hop count
Kaushik et al. [29,30]	Heterogeneous	Dynamic	Custom	Task directed graph	Multiobjective
Derin et al. [31]	Heterogeneous	Dynamic	ILP	Task and architecture graphs	Multiobjective
Zhe et al. [32]	Heterogeneous	Static	Artificial bee colony	Application task graph	Power consumption
Mandelli et al. [33,34]	Homogeneous	Dynamic	Custom	Application graph	Energy
Habibi et al. [35]	Heterogeneous	Static	Fuzzy and custom	Application characteristic graph	Message latency and energy
Huang et al. [18]	Heterogeneous	Static	ILP and SA	Task graph	Energy
Liu et al. [36]	Heterogeneous	Static	Ant colony	Task and core graphs	Energy and temperature
Zhong et al. [37]	Homogeneous	Static	Simulated annealing	Task graph	Energy
Rajaei et al. [21]	Heterogeneous	Static	Custom	Task graph	Energy
Sepulveda et al. [38]	Homogeneous	Hybrid	Multiobjective evolutionary	Application characterization graph	Latency and power
Sheng et al. [39]	Homogeneous	Static	Quadratic programming	Task graph	Energy
This work	Heterogeneous	Static or hybrid	PBIL	Task graph	Multiobjective

3. The proposed approach

As mentioned before, the proposed approach starts from an UML specification of the application. A tool chain is used in order to transform the high-level specification into an intermediate representation, which takes the form of a task graph. Such a task graph is used as the input of an optimization engine based on a PBIL algorithm, which is the responsible for the design space exploration. Finally, for the sake of providing some level of reliability to the system, a modified version of the PBIL algorithm is used to implement a dynamic mapping approach, which reacts as a consequence of a runtime failure. The three main elements just mentioned, i. e., the transformation tool chain, the optimization engine, and the dynamic mapping approach, are about to be described in this section.

3.1. The transformation tool chain

A high-level UML specification is used to describe the application in the early stages of the design process. Such specification contains only behavioral information and constraints of the design, as well as indications regarding data dependences among the executable objects of the system. On the other hand, the optimization engine starts from a common domain specification. Such specification combines the PIM information, which is present in the UML model, with specific platform annotations, which represent figures of merit derived from profiling and modeling. Fig. 1 depicts the process of converting the UML specification into a task graph, which represents the common domain specification, in the form of a TGFF file.

As can be seen in Fig. 1, the initial UML specification is converted in an Ecore metamodel, which is one of the two metamodels which support the EMF engine [40]. Such conversion is available in frameworks such as Eclipse [41] and Papyrus [42]. The metamodel is then converted in a task graph by means of the *Ecore2Groove* routine, which is part of the Groove tool set [43]. Finally, the resulting task graph is parsed in order to generate a TGFF file, which represents the input to the optimization engine. The specific-platform annotations are added at this stage.

In a formal way, the TGFF file represents an Annotated Directed Acyclic Graph (ADAG), which provides information regarding the data dependences among the system tasks, as well as implementation details related with the target architecture. Fig. 2 shows an instance of a given ADAG, where vertices represent executable tasks of the system, edges show the data dependences among such tasks, and annotations (which may be provided both for vertices as edges) provide some useful implementation details.

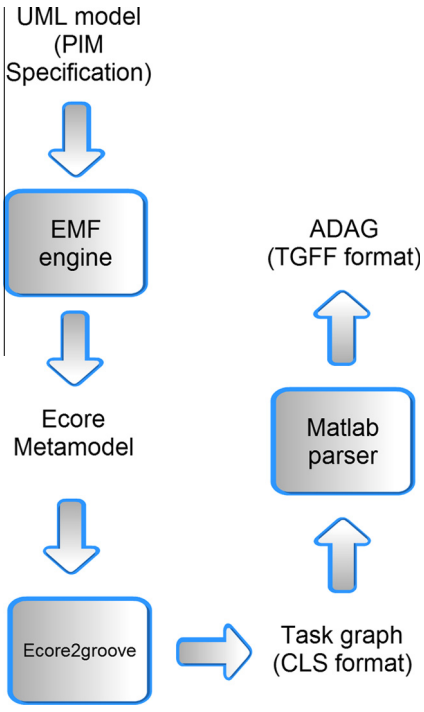


Fig. 1. Generation of the common domain representation, starting from an UML specification.

3.2. Mapping and scheduling of tasks

The PBIL optimization algorithm uses the information provided by the input ADAG, as the one depicted in Fig. 2, in order to per-

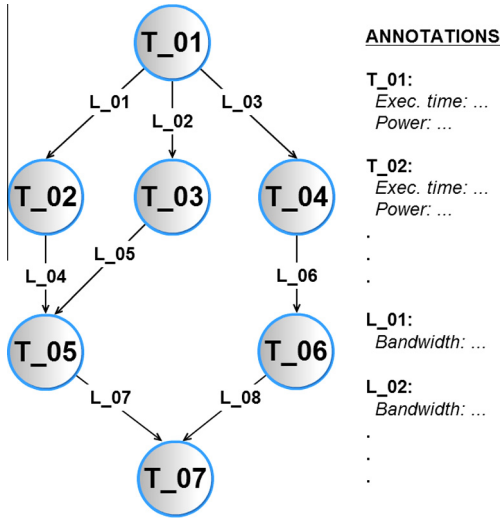


Fig. 2. A given ADAG generated by the code transformation engine.

form the design space exploration. Such exploration is aimed to find the best mapping and scheduling solution, according with some figures of merit. Although mapping and scheduling of tasks are two highly coupled processes they are going to be explained independently, for the sake of clarity.

3.2.1. Scheduling of tasks into a NoC

In a heterogeneous architecture, scheduling stage is responsible for finding an implementation resource (PE) for each task in the input ADAG. The placement of each of such resources is not resolved yet. The hardware platform offers a set of processing elements (PEs) with diverse features, so it is necessary to make choices regarding the implementation for each task of the application. Fig. 3 depicts a simple instance of a given scheduling performed starting from an input task graph with five tasks, which are going to be distributed over a set of four PEs. Since the target architecture is heterogeneous, there are several kind of PEs that may be potentially used in the tasks implementation. Fig. 3 shows generic microprocessors, digital signal processors, and IP modules. No previous decision has been made regarding the nature of the four PEs which are going to compose the NoC, neither regarding which tasks are going to be executed on each core. These choices are precisely the responsibility of the scheduling stage.

It has been supposed that tasks are served in a sequential fashion inside each PE. However, concurrency is allowed among tasks running in different cores. Although current timing schemas are much more complex, the former assumption allows predicting the starting and ending times for each task, which is the reason why this process is referred as the scheduling stage.

As can be seen in Fig. 3, the five tasks of the original ADAG are distributed among the four PEs of the architecture. Tasks *T*_01 and *T*_02 are scheduled to run in Core *A*, which is a generic microprocessor. Task *T*_03 is matched with Core *B* (a DSP), and so forth. Some tasks are executed concurrently, since each core is independent from each other in the system (as an example, tasks *T*_02 and *T*_03, which are running on different cores, are executed simultaneously). Scheduling of tasks affects directly those features of the system related with computation, such as execution time and power consumption. Nevertheless, some other features related with the communication among tasks are influenced also by this process.

The result of the scheduling stage may be represented by another graph, referred as core graph, in which each vertex represents a given core (or PE) of the system (which may have been intended to run several tasks). Edges in a core graph represent data dependences among cores. Left side of Fig. 4 shows the core graph resulting from the scheduling process depicted in Fig. 3.

3.2.2. Core mapping

Once the scheduling stage has defined a matching between tasks and resources, it is necessary to define the physical placement of each core in the NoC. This process is called core mapping or simply mapping. Fig. 4 shows an instance of the mapping process, which depicts the continuation of the scheduling stage of Fig. 3. As already mentioned, left side of Fig. 4 represents the input of the mapping stage, often referred as core graph. Right side of Fig. 4 represents the network architecture (a 2D mesh in this instance) in which the cores are going to be placed.

The core mapping stage has a direct impact in figures of merit related to communications among tasks, such as link delays, required bandwidth, traffic patterns, and so on. Features like power consumption and performance are also affected by the mapping stage, just because a given mapping solution may lead to high energy dissipation, related to the communication among tasks or unacceptable communication delays. After scheduling and core mapping, each task of the system is assigned to a given core, which in turns has a physical location in the network. Next subsection describes the proposed algorithm, which performs scheduling and core mapping by treating these processes as a whole optimization problem.

3.2.3. The PBIL algorithm

PBIL algorithms are stochastic search methods which use their best solutions, in order to achieve some directional information. Such approach has been used in embedded systems design automation with promising results [44,45]. The main feature of PBIL algorithms is the use of a population of potential solutions which

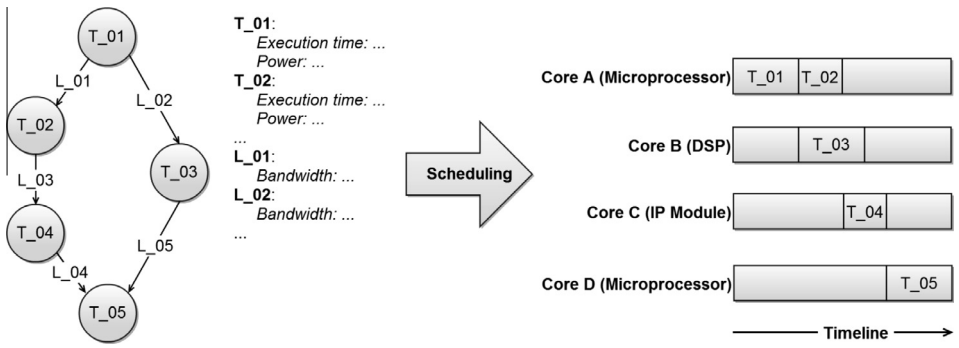


Fig. 3. Scheduling instance in a heterogeneous architecture.

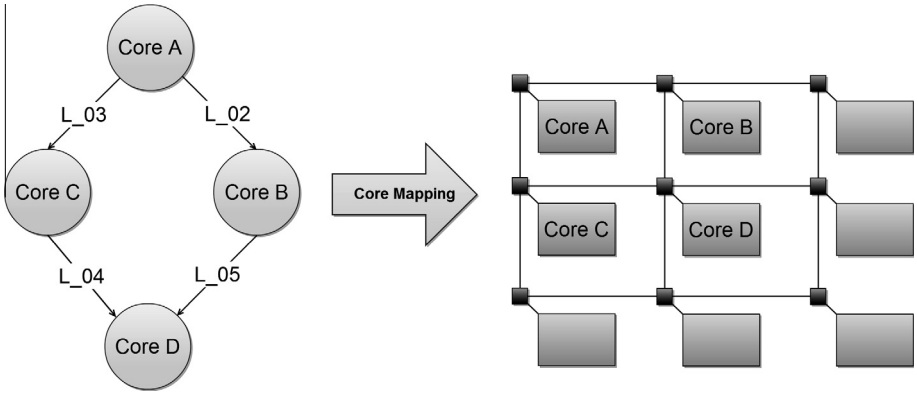


Fig. 4. Core mapping instance in a heterogeneous architecture.

converges toward an optimal. There are some other approaches based on population, such as genetic or evolutionary algorithms. The prominent difference of PBIL with respect to other population-based techniques is the way in which population is represented.

Fig. 5 shows an example of such representation, which is often referred as probability array. Let us suppose that an optimization problem may be defined by a set of N attributes or optimization queries. Each attribute may have up to M solution choices, so a given solution to the problem is completely defined as a set of N values in the range from 1 to M. Each column of Fig. 5 represents a single attribute of the optimization problem, and each entry of such a column corresponds to the probability of a choice to be used in the optimal solution. Since each attribute represents a conjoint probability variable, the sum of values along a single column in the probability array must be equal to one.

At the beginning of the PBIL algorithm, probability values in the array must be initialized to ensure maximum population diversity, i.e., if the amount of choices for a given attribute is equal to M, the values of the column associated with such an attribute must be initialized to $1/M$. The key after the initialization stage is to generate populations of potential solutions iteratively. The solutions at each iteration (generation) of the algorithm are generated from the current probability values in the PBIL array, i.e., if some entry in a given column has an increased value with respect to the remaining ones, its associated value shall appear more frequently in the individuals of the generated population. At each algorithm's iteration, the best solutions are chosen and used to increase the associated probabilities in the array.

	Attribute 1	Attribute 2	⋮	Attribute j	⋮	Attribute N
Choice 1	P(1,1)	P(1,2)	...	P(1,j)	...	P(1,N)
Choice 2	P(2,1)	P(2,2)	...	P(2,j)	...	P(2,N)
...	...	...	...	...	...	...
Choice i	P(i,1)	P(i,2)	...	P(i,j)	...	P(i,N)
...	...	...	...	...	...	...
Choice M	P(M,1)	P(M,2)	...	P(M,j)	...	P(M,N)

Fig. 5. A PBIL probability array.

Increasing some single value in a given column implies that the remaining values must be decreased in consequence. The way to perform such a probabilities update is a modified version of the Hebbian rule [46], as shown in Eq. (1). In Eq. (1), it is supposed that for a given attribute j, the best solution obtained is choice k. Suffixes *Old* and *New* in Eq. (1) are meant to denote the old and new versions of each probability, respectively.

$$P_{(i,j)_{New}} = \begin{cases} P_{(i,j)_{Old}} + (1 - P_{(i,j)_{Old}}) \times LR & \text{if } i = k \\ (1 - P_{(k,j)_{New}}) \times \frac{P_{(i,j)_{Old}}}{1 - P_{(k,j)_{Old}}} & \text{if } i \neq k \end{cases} \quad (1)$$

First part of Eq. (1) shows that if the current entry of the array (P_{ij}) resulted to be the associated with the optimized found solution, such probability in the PBIL array must be increased, by an amount that is proportional to the LR parameter. In the same sense, the remaining values of the actual column must be decreased by preserving of their relative weights. This is performed in the second part of Eq. (1).

The learning rate parameter, which is represented in Eq. (1) as *LR*, is the responsible of controlling the convergence speed of the PBIL algorithm. High values of *LR* may lead to quick convergence at expense of a low quality of the found optimal. Alternatively, low values of *LR* are related to a better solution-space exploration, which often implies higher convergence times. An adaptive version of the PBIL algorithm has been used in the proposed mapping and scheduling approach. The latter implies the change of the *LR* parameter of the Eq. (1), in order to speed up the convergence process. *LR* parameter allows to control the speed in which the probability array approaches a given optimal. To favoring space exploration (i.e., the process of searching over the overall solutions space), *LR* parameter must be kept at low values at the early iterations of the PBIL algorithm. When the algorithm approaches to a given optimal, the *LR* parameter may be increased for the sake of reaching a solution more quickly (this is often referred as space exploitation).

The PBIL array entropy is used in order to assess the status of the algorithm i.e., whether the algorithm is close or not to finding a given optimum. Entropy's calculation is performed in the same way as is done in information theory, as shown in Eq. (2). The value of E in Eq. (2) is maximal when all the values in the probability array are equal. As some entries of each column in the array become higher than the remaining ones, entropy value decreases. In the case in which a unique entry of each column of the array is equal to one (the remaining ones should be equal to zero), the associated entropy value will be zero.

$$E = -\frac{1}{N} \times \sum_{i=1}^{M} \sum_{j=1}^{N} P_{(i,j)} \times Log_M(P_{(i,j)}) \quad (2)$$

A learning rule is the way in which the *LR* and *E* parameters are related. A sigmoidal learning rule was used to improve the speed of convergence of the proposed PBIL algorithm [47]. Formalization of such a sigmoidal relationship is shown in Eq. (3).

$$LR = LR_{MIN} + \frac{LR_{MAX} - LR_{MIN}}{1 + e^{(2\Delta E - \Delta)}} \quad (3)$$

Algorithm 1 summarizes the basic PBIL optimization process. The meanings of the routines depicted in the algorithm, such as *Update_Array*, *Learning_Rule*, or *Entropy*, are straightforward, as described so far. The termination condition for the algorithm is given in terms of a tolerance, since waiting until de value of *E* becomes equal to zero is very prohibitive.

Algorithm 1: Basic PBIL algorithm

Input: An $M \times N$ probability matrix, called *P*
Output: An optimized solution for the problem at hand
begin
  $P(i,j) = \frac{1}{M}; \forall 1 \leqslant i \leqslant M$ and $1 \leqslant j \leqslant N$;
  repeat *Pop* = *Create_Population*(*P*);
    Fitness = *Evaluate_Population*(*Pop*);
    Best = *Choose_Best*(*Pop*,*Fitness*);
    E = *Entropy*(*P*);
    LR = *Learning_Rule*(*E*);
    P = *Update_Array*(*P*,*Best*,*LR*);
  until (*E* > *Tolerance*);
  return *Best*;
end

3.2.4. Mapping and scheduling by means of a PBIL algorithm

The first stage in using a PBIL algorithm for solving a given optimization problem, is the definition of a suitable PBIL array representation. Fig. 6 shows the proposed representation for the mapping and scheduling problem. As can be seen, such representation is composed by two matrices. The first matrix, which is placed in the left-anterior side of the figure, is used to determine the type of resource that must be placed on each tile of the NoC. The right-front side matrix determines the physical placement of each task of the system. Although the arrays depicted in Fig. 6 are not a direct representation of mapping and scheduling, yet provide a completely defined solution to both processes. As explained before, at the end of the optimization algorithm, each column of the matrices in Fig. 6 will have an entry which stands out from the others, meaning that a given choice is more probable to provide an optimal solution.

Arrays *P* and *Q* in Fig. 6, may be updated accordingly with the adaptive process depicted in Algorithm 1, with very slight modifications. Initialization of such arrays must take into account each array's dimensions, as explained before. Finally, the total entropy may be calculated as the mean of the entropies of both matrices.

There may be several objectives or figures of merit to optimize while conducting the design space exploration. A weighting vector and an aggregation approach have been used for the sake of combining the assessments related to such objectives. By changing the relative values present in the weighting vector, a designer may give more importance to some objectives with respect to the remaining ones. Several solutions with different tradeoffs among the optimization objectives may be generated in such a way. For the sake of avoiding that some algorithm executions aimed to generate different tradeoffs among the objectives, converge to the same optimal, a kernel approach based on distance [48] was used in this approach.

3.3. Reliability through hybrid mapping

The introduction of reliability features at the design stage is gaining relevance, as a consequence of three main facts [49]: nowadays systems are more prone to dissipate higher quantities of energy, current manufacturing process into the submicron scale are threatened by higher variability, and complexity in embedded systems is experiencing exponential rises. One of the most common approaches aimed to improving system reliability is fault tolerance, which is nothing but the ability of the system to provide correct service operation, even though some failures have taken place in the system. In turn, fault tolerance may be accomplished by means of redundancy.

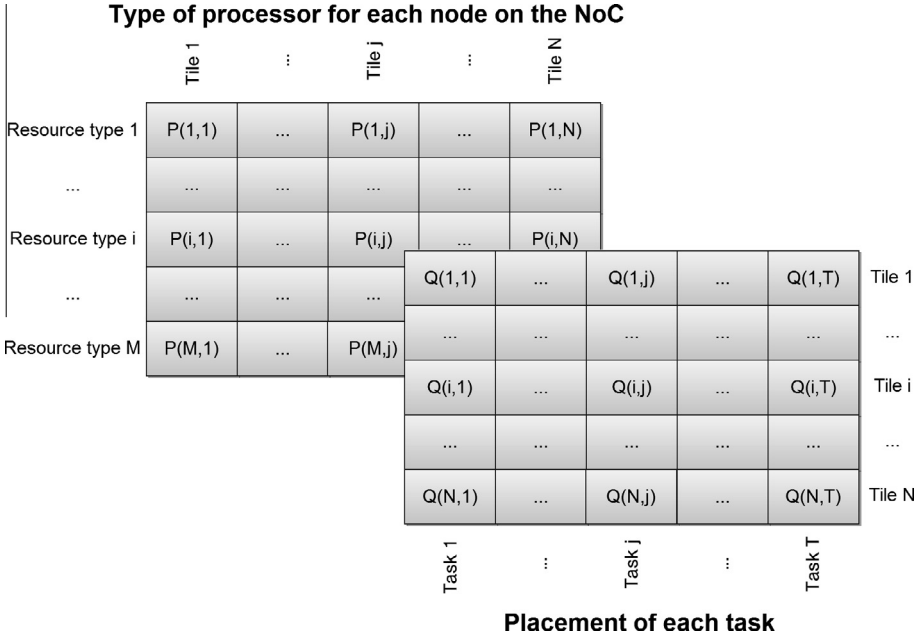


Fig. 6. Proposed PBIL array for mapping and scheduling.

There exist two kinds of redundancy: spatial and temporal. In spatial redundancy, several hardware independent resources are aimed to the execution of each task of the system, so if there is a failure event in some given resource, the remaining ones are able to provide a correct result. In contrast with spatial redundancy, temporal redundancy does not use several resources for the execution of a single task. Each task is mapped to a single hardware resource for its execution. If a failure event is detected, the execution stops and the tasks associated with the faulty PE are issued for their execution on a different resource. This means that in temporal redundancy, a task will be executed only on a single resource at a given time.

The mapping and scheduling of tasks as described so far are static in nature, since they are computed in design time. Dynamic mapping occurs at runtime and it refers to the process of reallocating tasks in the nodes of the NoC, as a consequence of two possible scenarios: in the first place, tasks may be reallocated to deal with changes in the operating conditions, such as the traffic, or the workloads in some nodes. Secondly, dynamic mapping may be used to deal with faulty nodes, in order to remap tasks to nodes which are operating correctly. In this work, we propose a temporal redundancy strategy, in order to provide some degree of fault tolerance. In terms of the nature of the mapping calculations, our approach may classified as hybrid, since part of the mapping calculations is performed in design time and the remaining mapping actions take place in runtime.

Some previous works suggest that Dynamic mapping calculations may last up the same amount of time used for static calculations [44]. In fact, in order to test the proposed dynamic solution, static mapping formulations are used as worst-case problems. The latter makes sense when considering that in a static mapping scenario the amount of resources and tasks to be mapped corresponds to the upper boundary of the same amounts in the dynamic problem (i.e., whereas dynamic mapping deals with a subset of tasks and resources, static mapping faces the whole set).

High time delays must be avoided in the dynamic mapping calculations, since such calculations take place in runtime. For such a reason, our mapping proposal relies on mapping solutions previously calculated in design time. The customized version of the PBIL algorithm, as the one described in Algorithm 1, is used for calculating mapping solutions to some defined failure scenarios, which shall be available for its use in runtime. Fig. 7 depicts such an approach.

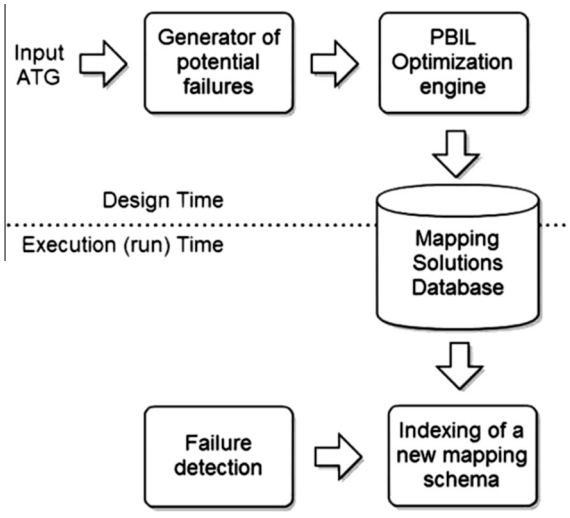


Fig. 7. Proposed fault tolerance approach.

Strictly, the mapping approach depicted in Fig. 7 must be classified as hybrid, since part of the mapping calculations are performed in design time and some other portion of the work takes place at runtime. Annotations provided in the input ADAG must include failure probabilities for each resource in the implementation platform, which is often the case in applications aimed to provide some level of reliability. By means of such failure probabilities, it is possible inferring some selected failure scenarios (i.e., those scenarios which are more prone to appear at runtime). The generator of potential failures feeds the PBIL optimization engine which is the responsible of finding remapping schemas in response to such given failures.

The PBIL algorithm may use a simple matrix representation for the finding of mapping schemas as a response to failure events, just because at runtime placing of resources in the NoC must have been already decided. This is the reason why the PBIL array for dynamic mapping takes the form of the matrix depicted in Fig. 5, instead of the two-matrix representation, which is shown in Fig. 6. In Fig. 5, the PBIL matrix has dimensions M by N, where N shall be equal to the amount of tasks that must be remapped and M shall be the amount of non-faulty resources, available for tasks execution. As already explained, each entry on the PBIL matrix, represents the probability of some task to be mapped on a given resource. The PBIL algorithm for dynamic mapping may then take the form described in Algorithm 1.

At the end of the optimization process, the mapping solution is represented by an integer-valued vector with N entries, where each of them represents an optimal mapping decision with respect to a given associated task. As a consequence, it can be said that the proposed mapping solution does not impose a meaningful storing overhead to the system, since the mapping solution to each failure scenario may be represented as a single vector. The amount of failure scenarios that should be taken into account in design time shall depend on the time that the designer wants to expend in the performing of pre-calculations, and, in lesser degree, on the storage constraints for saving the mapping solutions at runtime.

Both single-node and multiple-node failure scenarios may be considered in our remapping approach. Although next section shows only simulation results for single-node failure scenarios, the proposal described so far imposes no restriction to the nature of the failure scenario. Next section discusses some further considerations regarding this issue.

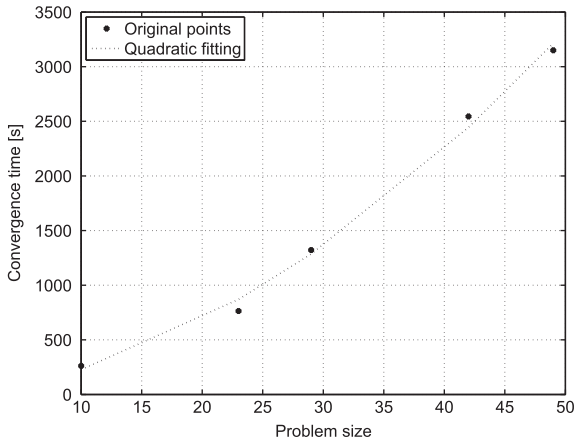
4. Experimental results

4.1. Static mapping and scheduling of tasks

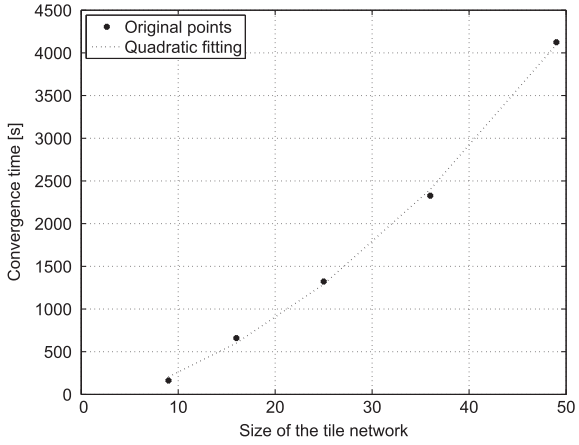
Several tests regarding synthetic and real-world mapping and scheduling problems were performed to the proposed approach, which was described in Section 3.2.4. For simulation purposes, the target architecture was set as a 2D mesh NoC with changeable size. A deterministic XY algorithm was used for simulating the routing of information across the network. The simulations were conducted by means of Matlab, which was running on a PC with a Core i7 processor and 8 Gbytes of RAM memory. A sigmoidal learning rule was used for implementing the adaptive behavior of the PBIL algorithm [47].

Regarding tests performed with synthetic problems, several convergence times were obtained with respect to changing values of both target NoC size and problem size (in terms of the amount of tasks in the input ADAG). Fig. 8 depicts the behavior of the convergence time as a function of the mentioned changing parameters.

Four optimization objectives were taken into account in the mapping and scheduling stage, namely, completion time, power consumption, mean number of hops for each message traveling



(a) Changing problem size.



(b) Changing network size.

Fig. 8. Behavior of the convergence time as a function of changing sizes of synthetic problems. (a) Changing problem size. (b) Changing network size.

across the network, and peak bandwidth on the set of communication links. Regarding synthetic problem simulations, there may be up to fifty resource types, which corresponds to the value of M in Fig. 6.

In Fig. 8.a, the abscissa represents the size of the problem, in terms of the number of tasks of the input ADAG, for a network of 5 by 5 tiles. Fig. 8.a shows a nearly quadratic behavior of the convergence time with respect to the size of the optimization problem, which makes sense when considering that the internal representation of the PBIL algorithm is based on two bidimensional matrixes, and the size of the input ADAG has a direct relationship to the dimensions of such matrixes. On the other hand, Fig. 8.b shows the dependence between the convergence time and the size of the tile network. Several instances of the algorithm were executed with the amount of tasks fixed and equal to thirty. The tile network size was changed from 9 (3×3 mesh) to 49 (7×7 mesh), leaving the rest of parameters unaltered. Again, there is a quadratic relationship between these two values, since the tile network size is one of the dimensions of the PBIL matrix representation. Similar tests were conducted regarding the dependence of convergence time with respect to the number of resource types. The quadratic behavior appeared again.

For the sake of performing further tests to the proposed mapping and scheduling algorithm, the Embedded System Synthesis Benchmarks Suite (E3S) was used. E3S is a widely used benchmark for testing of embedded systems [50–53]. The benchmark contains profiling information regarding real applications such as automotive industry, consumer, networking, telecom, and office automation. The implementation resources are composed by a set of up to seventeen PEs. Such set includes AMD, IBM Power PC and NEC processors, among others. Only the "cords" profiles were used, since they are the only ones which take into account traffic information. The remaining test conditions for the mapping and scheduling algorithm are the same described so far.

Table 2 depicts the behavior of the convergence time for several applications of the E3S suite. As shown in such a table, there is again a quadratic behavior regarding the relationship between convergence time and size of the network.

Although some of the convergence times depicted in Table 2 seem restrictive, two important facts must be taken into account, for the sake of using the proposed PBIL algorithm in real design scenarios:

- The mapping and scheduling stage is performed in design time, so a time surplus related to the optimization process is affordable. Due to the complexity related to the mapping stage in the NoC design process, it is necessary to establish an appropriate time budget for such a stage.
- The PBIL optimization algorithm was implemented in Matlab, by using a completely sequential coding. The convergence times are susceptible of improvement by using different development tools for the algorithm implementation or by parallelizing the algorithm itself. Regarding PBIL algorithm parallelization, a quick inspection performed to Algorithm 1 and Fig. 6 allows inferring that each column of the PBIL matrices may be processed independently. Then, the creation of new individuals, the entropy calculation, and the updating of the column probabilities, may be performed in a concurrent fashion, for the sake of speeding up of the convergence process.

Fig. 9 shows the evolution of the objective values, for the Automotive application in the E3S suite. The mapping and scheduling stage was performed on a 4×4 NoC, taking into account the same four objectives used in the previous section: completion time, power, number of hops, and peak bandwidth. As with previous instances of the mapping and scheduling optimization, the bandwidth values have been normalized with respect to a value of 100 MHz. Fig. 9 also shows how the objectives are decreased to their minimum values whilst the algorithm converges. Step-like behavior of some of the graphs is related to the drastic changes performed to the optimization objectives, as a consequence of a change in the optimal solution.

Table 2
Implementation results for the E3S benchmark.

Application	Amount of tasks	NoC size	Mean convergence time [s]
Auto-Indust	24	4×4	300.04
		5×5	421.28
		6×6	744.51
		7×7	1169.50
Consumer	12	4×4	265.64
		5×5	450.86
		6×6	750.72
		7×7	1305.97
Networking	13	4×4	196.85
		5×5	291.05
		6×6	441.29
		7×7	640.3
Telecom.	30	4×4	105.31
		5×5	243.51
		6×6	378.92
		7×7	615.52

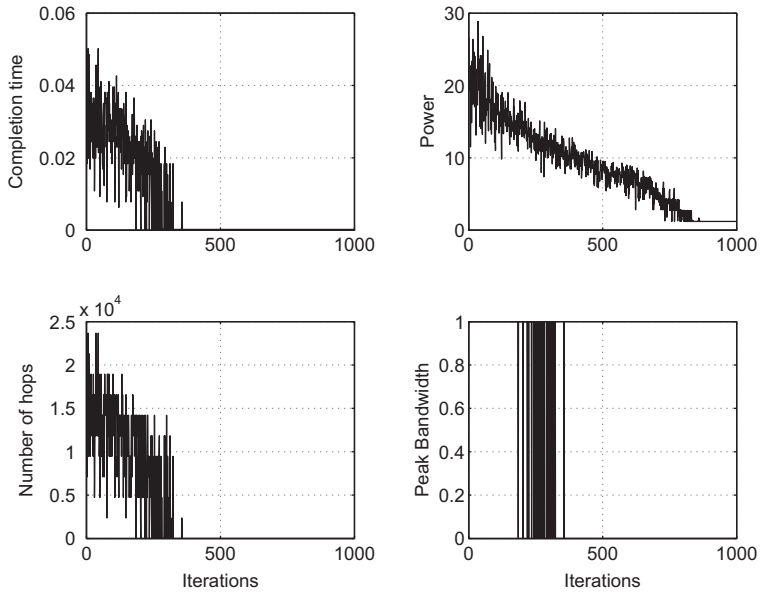


Fig. 9. Evolution of the optimization objectives for the E3S Benchmark.

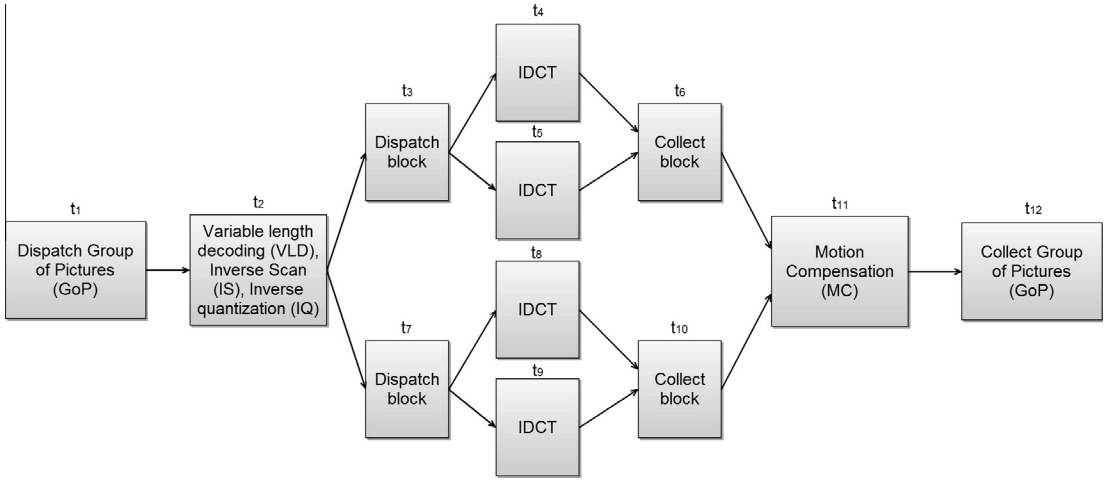


Fig. 10. A MPEG2 decoder representation.

Traffic among tasks, which is represented by the number of hops and required bandwidth in Fig. 9, depends on the application itself and also on the mapping schema chosen by the optimization algorithm. Different input task graphs will lead to different traffic patterns across the network. The mapping schema may change such patterns by grouping the communicating tasks in the application and then reducing the required bandwidth between nodes. In any case, by taking into account figures of merit related to communication features in the optimization stage, it is possible to deal with such features. As already mentioned, in our tests such figures of merit were the number of hops and the required peak bandwidth.

4.2. Remapping based on PBIL

Fig. 10 shows a parallelized representation of a MPEG video decoder [54], which was used for testing the proposed remapping approach. The figure shows how some of the processes related to MPEG decoding must be performed in a sequential way. Such is the case of the block which performs the dispatching of Groups of Pictures (GoPs) and the block which performs variable length decoding (VLD), Inverse Scan (IS), and Inverse Quantization (IQ). On the other hand, the Inverse Discrete Cosine Transform (IDCT) may be performed concurrently over different blocks of information, so four independent blocks have been devoted for such calculations.

Each processing block of Fig. 10 has been associated with a task of the input ADAG for the mapping algorithm based on PBIL. Labels have been added above each block for representing the corresponding task in the input ADAG. Several simulations were also performed for MPEG decoders represented by input ADAGs with sizes of 24 and 36 tasks.

For the sake of comparing the remapping results of the proposed solution with a previously reported work, the NoC target architecture used to evaluate the proposed dynamic mapping approach was a 3×3 2D mesh, composed by RISC and DSP processors, as shown in Fig. 11. In such a figure, there are nine nodes or tile spaces (labeled from $n1$ to $n9$) representing the processing elements and twelve communication links among the different nodes (labeled from $L1$ to $L12$). Placement of the PEs in the network has been previously decided by means of a static mapping as the one explained in Section 3. A deterministic XY routing algorithm was

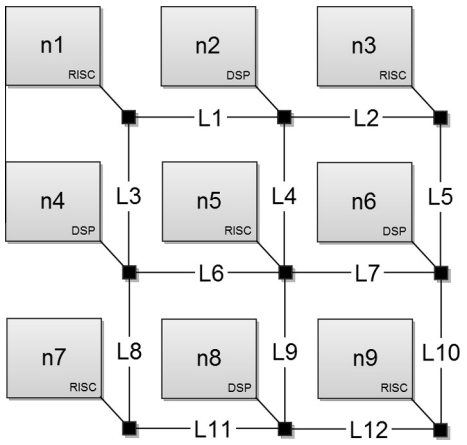


Fig. 11. Target NoC for the MPEG2 decoder.

used for simulating the routing of information into the network. The PBIL optimization was performed over two conflicting objectives: first, the completion time of the application, which may be calculated as the maximum time stamp associated with the execution of tasks in the whole system. The second optimization objective was the peak bandwidth of the target NoC. This figure of merit may be calculated as the maximum value of bandwidth requirements for links in the network, once the mapping has been performed.

According with previous discussion, the worst case scenario for the remapping of tasks agrees exactly with the static mapping problem, in which the whole set of resources and tasks must be taken into account. This approach was the adopted for the mapping tests performed with the MPEG2 decoder.

Fig. 12 shows the convergence times for the PBIL algorithm. As a reference, results of PBIL optimization are compared with those reported in [31]. In such a case, the optimization of the remapping schema was performed by means of an Integer Linear Programming (ILP) algorithm. As shown in Fig. 12, ILP approach exhibits a better performance than PBIL for small problems. However, if the number of tasks increases, optimization through ILP algorithm becomes prohibitive. Mean ILP convergence time for a 36-tasks input ADAG is around 1700 s. PBIL mean convergence time is around one order of magnitude below this value.

Quality of the remapping solutions may be measured in terms of the degradation caused to the optimization objectives, once

Table 3
Mean degradation for several executions of hybrid mapping algorithm.

Problem size (number of tasks)	Completion time degradation [%]	Bandwidth degradation [%]
12	0	6
24	0.17	20.02
36	2.83	0.90

the tasks have been reallocated to the non-faulty resources. Best dynamic mapping solutions can deal with node failures, without demoting the original values of the objectives (which were already optimized in the static mapping stage). Table 3 shows the mean relative degradation obtained from the remapping of the tasks with respect to the original static mapping solution, as a result of a single node failure. As can be seen, there is a mean degradation of 5% of the optimization objectives as a consequence of the remapping of the tasks.

After several executions of the PBIL algorithm for dynamic mapping, it could be observed that failures on node 5 ($n5$ in Fig. 11) are much more severe than for the remaining nodes. When there is a failure in node 5, dynamic mapping may lead to degradations up to 90%. Node 5 is also present in all the optimal solutions obtained from the static mapping process. By excluding this critical node of the set of potentially faulty resources, as is done in [31], degradation caused as a consequence of remapping, may be drastically improved. This behavior may be explained by realizing that the $n5$ node has an important position in the network, and it represents the only RISC node which is connected directly to the four available DSPs. It is logical that in the static mapping stage, a significant number of tasks were allocated to such a critical node. In other words, given its position and the nature of its connections in the network, node $n5$ is unsuitable for temporal redundancy in a failure event. Therefore, different fault tolerance strategies, such as the use of static redundancy should be considered for this node.

In the same sense, some failure scenarios may be impractical or may lead to paradoxical situations. For instance, let us consider a scenario in which nodes 2 and 4 ($n2$ and $n4$) of Fig. 11 have failures, whilst node 1 ($n1$) is working correctly. The generator of potential failures of Fig. 7 must include a correction mechanism, in such a way that the PBIL algorithm discards the node $n1$ as a potential mapping target, and treat it as an additional faulty resource.

Finally, by changing the relative values of the weighting vector, already described, it was possible to construct a near-Pareto curve, which relates the two optimization objectives considered in the MPEG2 remapping problem. Fig. 13 depicts the curve derived from

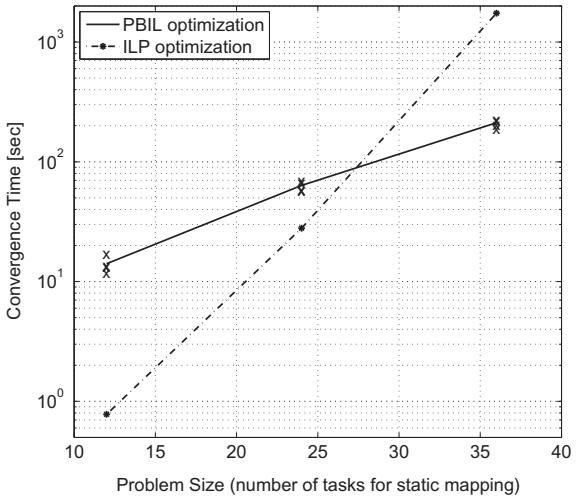


Fig. 12. Comparison between PBIL and ILP remapping optimization.

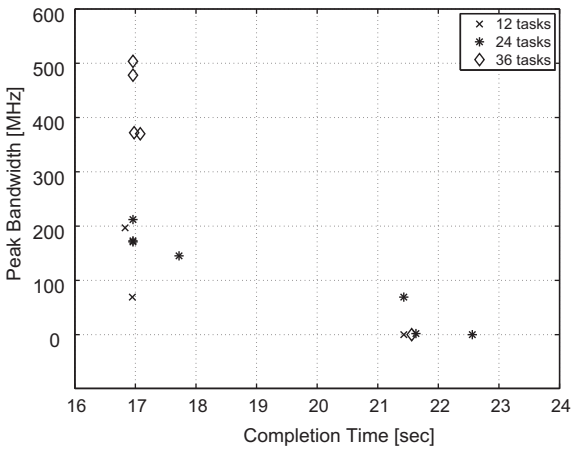


Fig. 13. A near-Pareto curve constructed for the MPEG2 problem.

several simulations for different problem sizes. The weighting strategy allows obtaining several tradeoff solutions among the optimization objectives, which may be an appealing feature for the designer.

5. Conclusions

A mapping and scheduling solution has been described and tested, using as target architecture a NoC, and starting from a UML specification. The mapping and scheduling problems have been treated as a conjoint optimization problem, and solved by means of a PBIL algorithm. A fault tolerance approach, based on the PBIL optimization algorithm was also presented.

With respect to the adaptive PBIL optimization algorithm, the promising results suggest that it is a better tradeoff in terms of performance, when compared with other reported solutions, such as ILP. Adaptive PBIL algorithm seems to be very suitable for dealing with complex multiobjective problems.

Regarding the mapping of tasks in runtime, the performance of the PBIL approach was assessed in terms of both convergence time and mean degradation of the optimization objectives, as a consequence of the remapping strategy for dealing with node failures. Convergence times resulted to be better for problems of higher size, when compared with reported results with ILP optimization. With respect to the mean degradation of the objectives, it was up to 20% with respect to the original optimized values.

Acknowledgements

The authors thank to ARTICA, to COLCIENCIAS, to ICT Ministry of Colombia, to National University of Colombia and to University of Antioquia, for their support in the development of this work.

References

[1] Alberto Sangiovanni-Vincentelli, Quo vadis SLD: reasoning about trends and challenges of system-level design, Proc. IEEE 95 (3) (2007) 467–506.
[2] OMG, UML 2.0 Superstructure Specification, Technical Report, Object Management Group (OMG), August 2005.
[3] Thomas A. Henzinger, Joseph Sifakis, The embedded systems design challenge, in: Jayadev Misra, Tobias Nipkow, Emil Sekerinski (Eds.), FM, Lecture Notes in Computer Science, vol. 4085, Springer, 2006, pp. 1–15.
[4] Alberto Sangiovanni-Vincentelli, Is a unified methodology for system-level design possible?, IEEE Des Test 25 (4) (2008) 346–357.
[5] Sanford Friedenthal, Alan Moore, Rick Steiner, A Practical Guide to SysML: Systems Modeling Language, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2008.
[6] Safouan Taha, Ansgar Radermacher, Sébastien Gérard, Jean-Luc Dekeyser, MARTE: UML-based hardware design from modelling to simulation, in: FDL, ECSI, 2007, pp. 274–279.
[7] Jerome Hugues, Bechir Zalila, Laurent Pautet, Fabrice Kordon, From the prototype to the final embedded system using the ocarina AADL tool suite, ACM Trans. Embedded Comput. Syst. 7 (4) (2008) 42:1–42:25.
[8] Robert P. Dick, David L. Rhodes, Wayne Wolf, Tgff: Task Graphs for Free, 1998.
[9] Pavel Ghosh, Arunabha Sen, Alexander Hall, Energy efficient application mapping to NoC processing elements operating at multiple voltage levels, in: Proceedings of the 2009 Third ACM/IEEE International Symposium on Networks-on-Chip, NOCS '09, IEEE Computer Society, Washington, DC, USA, 2009, pp. 80–85.
[10] Ashwini Raina, Venkatesan Muthukumar, Traffic aware scheduling algorithm for Network on Chip, in: Proceedings of the 2009 Sixth International Conference on Information Technology: New Generations, ITNG '09, IEEE Computer Society, Washington, DC, USA, 2009, pp. 877–882,.
[11] Philip K.F. Hölzenspies, Johann L. Hurink, Jan Kuper, Gerard J.M. Smit, Run-time spatial mapping of streaming applications to a heterogeneous multi-processor system-on-chip (MPSoCs), in: Proceedings of the Conference on Design, Automation and Test in Europe, DATE '08, ACM, New York, NY, USA, 2008, pp. 212–217.
[12] Guangyu Chen, Feihui Li, S.W. Son, M. Kandemir, Application mapping for chip multiprocessors, in: Proceedings of the 45th Annual Design Automation Conference, DAC '08, ACM, New York, NY, USA, 2008, pp. 620–625.
[13] Changjiu Xian, Yung-Hsiang Lu, Zhiyuan Li, Energy-aware scheduling for real-time multiprocessor systems with uncertain task execution time, in: Proceedings of the 44th annual Design Automation Conference, DAC '07, ACM, New York, NY, USA, 2007, pp. 664–669.
[14] Ruibin Xu, Rami Melhem, Daniel Mosse, Energy-aware scheduling for streaming applications on chip multiprocessors, in: Proceedings of the 28th IEEE International Real-Time Systems Symposium, RTSS '07, IEEE Computer Society, Washington, DC, USA, 2007, pp. 25–38.
[15] Chuan-Yue Yang, Jian-Jia Chen, Tei-Wei Kuo, Lothar Thiele, An approximation scheme for energy-efficient scheduling of real-time tasks in heterogeneous multiprocessor systems, in: Proceedings of the Conference on Design, Automation and Test in Europe, DATE '09, 3001 Leuven, Belgium, Belgium, European Design and Automation Association, 2009, pp. 694–699.
[16] Michel Goraczko, Jie Liu, Dimitrios Lymberopoulos, Slobodan Matic, Bodhi Priyantha, Feng Zhao, Energy-optimal software partitioning in heterogeneous multiprocessor embedded systems, in: Proceedings of the 45th annual Design Automation Conference, DAC '08, ACM, New York, NY, USA, 2008, pp. 191–196.
[17] Amit Kumar Singh, Akash Kumar, Thambipillai Srikanthan, A hybrid strategy for mapping multiple throughput-constrained applications on MPSoCs, in: Proceedings of the 14th International Conference on Compilers, Architectures and Synthesis for Embedded Systems, CASES '11, ACM, New York, NY, USA, 2011, pp. 175–184.
[18] Jia Huang, Christian Buckl, Andreas Raabe, Alois Knoll, Energy-aware task allocation for network-on-chip based heterogeneous multiprocessor systems, in: Yiannis Cotronis, Marco Danelutto, George Angelos Papadopoulos (Eds.), PDP, IEEE Computer Society, 2011, pp. 447–454.
[19] Dimitris Bertsimas, Omid Nohadani, Robust optimization with simulated annealing, J. Global Optim. 48 (2) (2010) 323–334.
[20] N.M. Pindoriya, S.N. Singh, Kwang Y. Lee, A comprehensive survey on multi-objective evolutionary optimization in power system applications, in: Power Engineering Society, IEEE General Meeting, 2010, pp. 1–8.
[21] Ramin Rajaei, Shaahin Hessabi, Bijan Vosoughi Vahdat, An energy-aware methodology for mapping and scheduling of concurrent applications in MPSoC architectures, in: 19th Iranian Conference on Electrical Engineering (ICEE), May 2011, pp. 1–6.
[22] Wooyoung Jang, David Z. Pan, A3MAP: architecture-aware analytic mapping for networks-on-chip, ACM Trans. Des. Autom. Electron. Syst. 17 (3) (2012) 26:1–26:22.
[23] Eduardo Antunes, Matheus Soares, Alexandra Aguiar, Sergio Johann Filho, Marcos Sartori, Fabiano Hessel, César A.M. Marcon, Partitioning and dynamic mapping evaluation for energy consumption minimization on NoC-based MPSoC, in: Keith A. Bowman, Kamesh V. Gadepally, Pallab Chatterjee, Mark M. Budnik, Lalitha Immaneni (Eds.), ISQED, IEEE, 2012, pp. 451–457.
[24] Eduardo Antunes, Alexandra Aguiar, F. Sergio Johann, Marcos Sartori, Fabiano Hessel, César Marcon, Partitioning and mapping on NoC-Based MPSoC: an energy consumption saving approach, in: Proceedings of the Fourth International Workshop on Network on Chip Architectures, NoCArc '11, ACM, New York, NY, USA, 2011, pp. 51–56.
[25] Ou He, Sheqin Dong, Wooyoung Jang, Jinian Bian, David Z. Pan, UNISM: unified scheduling and mapping for general Networks on Chip, IEEE Trans. VLSI Syst. 20 (8) (2012) 1496–1509.
[26] Mohammad Hosseinabady, Jose Luis Nunez-Yanez, Run-time stochastic task mapping on a large scale Network-on-Chip with dynamically reconfigurable tiles, IET Comput. Digital Tech. 6 (1) (2012) 1–11.
[27] Parisa Khadem Hamedani, Shaahin Hessabi, Hamid Sarbazi-Azad, Natalie D. Enright Jerger, Exploration of temperature constraints for thermal aware mapping of 3D Networks on Chip, in: Rainer Stotzka, Michael Schiffers, Yannis Cotronis (Eds.), PDP, IEEE, 2012, pp. 499–506.
[28] Chao Wang, Licheng Yu, Li Liu, Tianzhou Chen, Packet triggered prediction based task migration for Network-on-Chip, in: Proceedings of the 2012 20th Euromicro International Conference on Parallel, Distributed and Network-based Processing, PDP '12, IEEE Computer Society, Washington, DC, USA, 2012, pp. 491–498,.
[29] Samarth Kaushik, Amit Kumar Singh, Wu Jigang, Thambipillai Srikanthan, Run-time computation and communication aware mapping heuristic for NoC-based heterogeneous MPSoC platforms, in: Proceedings of the 2011 Fourth International Symposium on Parallel Architectures, Algorithms and Programming, PAAP '11, IEEE Computer Society, Washington, DC, USA, 2011, pp. 203–207.
[30] Amit Kumar Singh, Thambipillai Srikanthan, Akash Kumar, Wu Jigang, Communication-aware heuristics for run-time task mapping on NoC-based MPSoC platforms, J. Syst. Archit. 56 (7) (2010) 242–255.
[31] Onur Derin, Deniz Kabakci, Leandro Fiorin, Online task remapping strategies for fault-tolerant network-on-chip multiprocessors, in: Proceedings of the Fifth ACM/IEEE International Symposium on Networks-on-Chip, NOCS '11, ACM, New York, NY, USA, 2011, pp. 129–136.
[32] Li Zhe, Ling Xiang, Noc mapping based on chaos artificial bee colony optimization, in: International Conference on Computational Problem-Solving (ICCP), Oct 2011, pp. 518–521.
[33] Marcelo Mandelli, Luciano Ost, Everton Carara, Guilherme Guindani, Thiago Gouvea, Guilherme Medeiros, Fernando Gehm Moraes, Energy-aware dynamic task mapping for NoC-based MPSoCs, in: ISCAS, IEEE, 2011, pp. 1676–1679.
[34] Marcelo Mandelli, Alexandre Amory, Luciano Ost, Fernando Gehm Moraes, Multi-task dynamic mapping onto NoC-based MPSoCs, in: Proceedings of the 24th symposium on Integrated circuits and systems design, SBCCI '11, ACM, New York, NY, USA, 2011, pp. 191–196.
[35] Amirali Habibi, Mouhammad Arjomand, Hamid Sarbazi-Azad, Multicast-aware mapping algorithm for on-chip networks, in: Proceedings of the 2011 19th International Euromicro Conference on Parallel, Distributed and

Network-Based Processing, PDP '11, IEEE Computer Society, Washington, DC, USA, 2006, pp. 455–462,.
[36] Yanhua Liu, Ying Ruan, Zongsheng Lai, Weiping Jing, Energy and thermal aware mapping for mesh-based NoC architectures using multi-objective ant colony algorithm, in: Third International Conference on Computer Research and Development (ICCRD), vol. 3, March 2011, pp. 407–411.
[37] Liulin Zhong, Jiayi Sheng, Ming'e Jing, Zhiyi Yu, Xiaoyang Zeng, Dian Zhou, An optimized mapping algorithm based on simulated annealing for regular NoC architecture, in: IEEE Ninth International Conference on ASIC (ASICON), Oct. 2011, pp. 389–392.
[38] J. Sepulveda, M. Strum, Wang Jiang Chau, G. Gogniat. A multi-objective approach for multi-application NoC mapping, in: IEEE Second Latin American Symposium on Circuits and Systems (LASCAS), Feb. 2011, pp. 1–4.
[39] Jiayi Sheng, Liulin Zhong, Ming'e Jing, Zhiyi Yu, Xiaoyang Zeng, A method of quadratic programming for mapping on NoC architecture, in: 2011 IEEE Ninth International Conference on ASIC (ASICON), Oct. 2011, pp. 200–203.
[40] David Steinberg, Frank Budinsky, Marcelo Paternostro, Ed Merks, EMF: Eclipse Modeling Framework 2.0., second ed., Addison-Wesley Professional, 2009.
[41] Gabor Bergmann, Abel Hegedus, Akos Horvath, Istvan Rath, Zoltan Ujhelyi, Daniel Varro, Implementing efficient model validation in emf tools, in: Proceedings of the 2011 26th IEEE/ACM International Conference on Automated Software Engineering, ASE '11, IEEE Computer Society, Washington, DC, USA, 2011, pp. 580–583.
[42] Sébastien Gérard, Cédric Dumoulin, Patrick Tessier, Bran Selic, Papyrus: a UML2 tool for domain-specific language modeling, in: Proceedings of the 2007 International Dagstuhl Conference on Model-based Engineering of Embedded Real-time Systems, MBEERTS'07, Springer-Verlag, Berlin, Heidelberg, 2010, pp. 361–368.
[43] A. Rensink, The groove simulator: a tool for state space generation, in: J.L. Pfaltz, M. Nagl, B. Böhlen (Eds.), Applications of Graph Transformations with Industrial Relevance (AGTIVE), Lecture Notes in Computer Science, vol. 3062, Springer Verlag, Berlin, 2004, pp. 479–485.
[44] Freddy Bolanos, Jose Edison Aedo, Fredy Rivera, Nader Bagherzadeh, Mapping and scheduling in heterogeneous NoC through population-based incremental learning, J. Univ. Comput. Sci. 18 (7) (2012) 901–916.
[45] Le-jun Fan, Bin Li, Zhen-quan Zhuang, Zhong-qian Fu, An approach for dynamic hardware/software partitioning based on DPBIL, Proceedings of the Third International Conference on Natural Computation – ICNC '07, vol. 05, IEEE Computer Society, Washington, DC, USA, 2007, pp. 581–585.
[46] Ray H. White, Original contribution: competitive Hebbian learning: algorithm and demonstrations, Neural Netw. 5 (2) (1992) 261–275.
[47] Freddy Bolanos, Jose Edison Aedo, Fredy Rivera, Comparison of learning rules for adaptive population-based incremental learning algorithms, in: Hamid R. Arabnia, Ashu M.G. Solo (Eds.), ICAI, CSREA Press, 2012, pp. 244–251.
[48] B.W. Silverman, Density estimation for statistics and data analysis/B.W. Silverman, Chapman and Hall, London; New York, 1986.
[49] Shekhar Borkar, Norman P. Jouppi, Per Stenstrom, Microprocessors in the era of terascale integration, in: Proceedings of the Conference on Design, Automation and Test in Europe, DATE '07, San Jose, CA, USA, EDA Consortium, 2007, pp. 237–242.
[50] Jingcao Hu, Application-specific buffer space allocation for Networks-on-Chip router design, in: Proc. of the IEEE/ACM Intl. Conf. on Computer-aided Design, 2004, pp. 7–11.
[51] Jeremy Chan, Sri Parameswaran, NoCEE: Energy macro-model extraction methodology for Network on Chip routers, in: Proceedings of the IEEE/ACM International Conference on Computer Aided Design (ICCAD), 2005, pp. 254–259.
[52] Umit Y. Ogras, Radu Marculescu, Hyung Gyu Lee, Naehyuck Chang, Communication architecture optimization: making the shortest path shorter in regular networks-on-chip, in: Proceedings of the Design Automation and Test in Europe Conference, 2006, pp. 712–717.
[53] Umit Y. Ogras, Radu Marculescu, Puru Choudhary, Diana Marculescu, Voltage-frequency Island Partitioning for GALS-based Networks-on-Chip, in: PROC. DAC, 2007.
[54] Piero A. Bonatti, Carsten Lutz, Aniello Murano, Moshey Vardi, ISO IEC 13818-2 MPEG-2, information technology – generic coding of moving pictures and associated audio information: video, in: ICALP 2006, LNCS, Springer, 2006, pp. 540–551.

Freddy Bolanos is a professor from the National University of Colombia. Currently, he is a Ph.D. candidate from University of Antioquia. His research interests are in design methodologies for digital and embedded systems, computational tools for design automation and digital signal processing.

Fredy Rivera is currently with the Computer Science Department at the University of Antioquia, Colombia. He holds a Ph.D. from the Complutense University of Madrid, Spain. His research interests are in the embedded systems design and reconfigurable hardware. He works as a researcher in the Center of Excellence ARTICA.

Jose Edinson Aedo is a professor of the University of Antioquia. He is also the scientific coordinator of the research Center of Excellence ARTICA. He holds a Ph.D. degree in Electrical Engineering from the University of Sao Paulo, Brazil. His research interests include Parallel processing, Computer architecture, Integrated Circuits Design, Embedded Systems, Tools for Fast Prototyping in Hardware Design, and Power and Industrial Electronics.

Nader Bagherzadeh is a professor in University of California, Irvine. He holds a Ph.D. degree from the University of Austin, Texas (USA). Dr. Bagherzadeh is interested in low-power and embedded digital signal processing, computer architecture, computer graphics and VLSI design. Within the area of embedded digital signal processing, Dr. Bagherzadeh is interested in the design and VLSI development of reconfigurable processor architectures and their algorithm mapping for high-performance and low-power applications in mobile communications. This technology can be used for 3G and 4G cellular phones, as well as other telecommunications systems. In the area of computer graphics, Dr. Bagherzadeh has been involved in the development of a new scheme for creating computer-generated three-dimensional models of a scene based on previously recorded images captured with a standard digital video camera. These models can be used for military and civilian simulator applications, as well as movie special effects.