Multi-objective Mapping for NoC Architectures

Abou El Hassan BENYAMINA
Université d'ORAN ES-SENIA
BP 1524 EL mnaouer ORAN Algérie
benyanabou@yahoo.fr
abou-el-hassan.Benyamina@lifl.fr

Pierre BOULET
Université des Sciences et Technologies de Lille, cité scientifique, 59655 Villeneuve d'Ascq cedex, France
pierre.boulet@lifl.fr

Journal of Digital Information Management

ABSTRACT: *In order to improve the performance of current embedded systems, Network -on-Chip (NoC) offers many advantages, especially in terms of flexibility and low cost. Applications require more and more intensive computations, especially multimedia applications such as video encoding.*

Developing product and application using such an architecture offers many challenges and opportunities. Many tools will be required to develop a NoC architecture for a specific application. A tool which can map and schedule an application or a set of applications to a given NoC architecture will be essential and must be able to satisfy many relative trade-offs (real-time, performance, low power consumption, time to market, re-usability, cost, area, etc).

In this paper, we survey most approaches used to solve this problem with different goals. We then describe our approach based on a genetic algorithm and its implementation and its design objective.

Key words: Network-on-Chip, mapping objective, MPSoC, optimization, genetic algorithms

1. Introduction

As silicon technology keeps scaling, it is becoming technically feasible to integrate entire and complex systems on the same silicon die. This solution provides scalable computation power, and it is expected that many hundreds of processor cores will be integrated on these Multi-Processor Systems-on-Chip (MPSoCs) in future technologies. MPSoCs are widely used in embedded systems (such as cellular phones, automotive control engines, etc.) where, once deployed in field, they always run the same set of applications. An optimal allocation and scheduling for such applications can be statically derived off-line [13].

A critical task for recent MPSoCs is the minimization of the energy consumed. We start from a well-characterized task graph, a directed acyclic graph representing a functional abstraction of the application that will run on the MPSoC. Each task is characterized by the number of clock cycles used for its execution. Clearly the duration of each task and the energy spent for running it depends on the clock frequency used during the task execution. In addition, tasks connected by arcs in the task graph communicate if they are allocated to different processors. Each edge is characterized by a number that corresponds to the total number of bytes exchanged between two tasks.

Defining the optimal allocation, scheduling and voltage scaling for minimizing energy in MPSoCs is the aim of this paper. Energy is consumed during task execution and task communication.

The problem we face is very complex. Because we solve in the same time the allocation of tasks to the processors and find the optimal path allocation (or communication mapping) referred in literature as Network Assignment [7]. For this we have used two methods one based on genetic algorithms and other on Dijkstra's algorithm. The solution must also verify some constraints such area, memory, load balancing, link speed, bandwidth and certainly hard real time constraints.

1.1 NoC/MPSoC Design Flow

NoC/MPSoC design methodology is platform based; relying on IP core reuse and specialization of the communication structure (Figure 1) [1]. The first step is a generic NoC description which specifies the general features of the architecture such as topology, constraints on sizes of IP cores, communication principles, etc. The next design step is to specialize the generic architecture for an application or a class of applications by deciding the size of NoC, selecting the IP cores and finalizing the design of switches (routing algorithm) and protocol format.

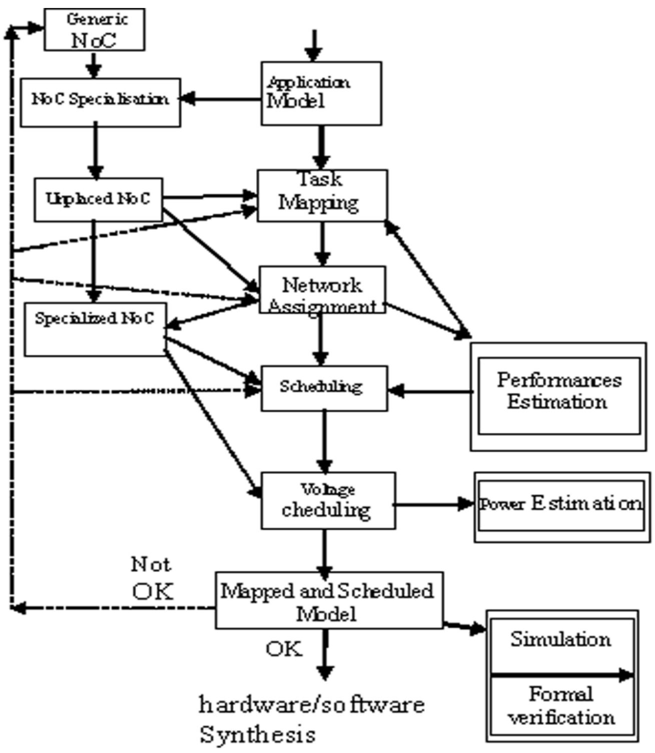


Figure 1. NoC Design Flow[1]

Mapping and Scheduling steps follow NoC Specialization and their role is to implement the given application in to the selected architecture which mainly means to assign and order the tasks and communications of the application into the resources of the architecture optimizing the design goals. Communication Mapping (CM) and Communication Scheduling (CS) raise more problems in NoC than in SoC because a minimal routing path must be allocated for each NoC communication, the exclusive usage of communication environment must be ensured without an arbiter and deadlock and congestion must be prevented. Moreover, in large NoC the communication distance has a big impact on communication delay.

After the NoC specialization and mapping steps, estimation is performed to foresee the feasibility and quality of the final solution using average or statistical measurements. After scheduling, the feasibility and the quality of the mapped and scheduled model are checked via simulations or formal analysis, such that the unfeasible solutions are replaced or new architectures tried.

1.2 NoC Design Goals

NoC/MPSoC design methodologies share many design goals with SoC design methodologies [1], namely, reducing energy consumption, minimizing the chip area and maximizing the timing performance. Energy saving is very important, especially in the design of portable embedded systems, where extended and correct functionality depends on the battery life time and circuit heating. Chip area is related to switch design and on-chip memories, whose layouts can occupy up to 80% from the total area. Reducing functional complexity of switches (routing algorithm) and maximizing the utilization of memories can reduce chip area. Timing constraints refers to hard and soft deadlines, whose misses could lead to failure or to quality degradation of the results. These are contradictory goals because minimizing power consumption implies to slow down the computations and thus affecting the system performance. Task mapping and NoC specialization can be targeted to any design goal, such as energy consumption, chip area and timing performances or to a combination of them. Thus, during NoC specialization, certain number and types of IP cores are selected depending on their cost and performances,

2. Mapping and Scheduling Related Issues

An important step of NoC design flow is Mapping and scheduling which deal with the implementation of the application on a specialized architecture. The inputs to the mapping and scheduling problem are the model of application(s), the model of target architecture, the performance and power constraints and the objective functions to be optimized.

The output of this step is a partitioning of the application(s) among computing resources on the platform and a schedule for the execution of the various computational tasks on these resources.

2.1 Architectural and Application Model

The architecture model is generally specified as a directed graph with two types of nodes representing the processing elements (PE) and switches which are interconnected by edges representing the communication links (CL) of the platform.

Several concurrent applications could be represented as a set of task graphs which could be executed in parallel on the computing platform. There are few parameters which enclose the representation: tasks execution times (ET) on each PE, hard and soft deadlines of tasks, task graph period, communication volume of each edge, energy consumption per processing cycle at nominal supply voltage, energy consumption per communication unit (bit/byte) with ignoring communication distance, memory requirements of tasks.

Figure 2 shows an example of mapping and scheduling problem for a 2D mesh topology NoC architecture with 3x3 resources. In the example, there are two concurrent applications represented by two task graphs. Mapping is defined as the assignment of tasks to processing elements and communications to communication routes. Mapping for NoC could also include the assignment of IP cores to NoC tiles, which together with routing path allocation (communication mapping) is referred in literature as network assignment. Network assignment is usually performed after task mapping and aims to reduce on-chip inter-communication distance.

Scheduling is the time ordering of tasks and communications on their assigned resources, which assures the mutual exclusion between executions of tasks on the same resource.

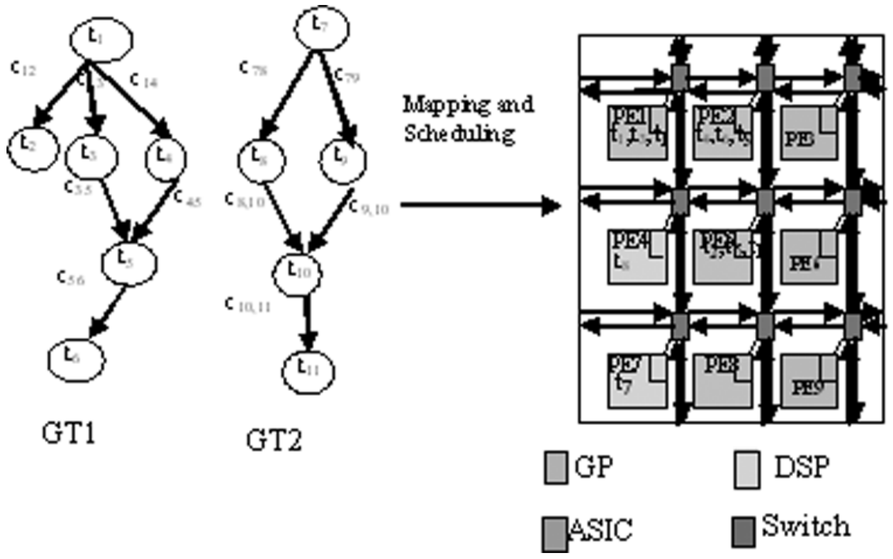


Figure 2. Illustration of Mapping and Scheduling Problem

2.2 Constraints and Objective Functions

A feasible mapping and scheduling is one which meets the design constraints such as timing constraints, data precedence constraints, memory size constraints etc. A quality mapping and scheduling is one which, besides meeting the design constraints, also optimizes the design goals such as minimizing energy consumption, maximizing timing performances or balancing memory utilization. Mapping and scheduling must, sometimes, make a trade-off between contradictory design goals and constraints such as reducing energy consumption and verifying hard real time constraints, minimizing communication volume and maximizing computation energy savings, maximizing memory utilization and verifying hard real-time constraints, etc.

Mathematical programming provides methods for minimizing or maximizing objective functions with constraints imposed on the variables of the problem.

Integer linear programming (ILP), non-linear programming (NLP) and mixed integer linear programming (MILP) are examples of mathematical programming. Constraint linear programming (CLP) is the upper class of ILP and MILP. ILP and MILP are problems were objective and constraints functions are linear and all variables or only some of them are integers. NLP are problems were objective or constraint functions are non-linear. MILP is NP-complete, while for ILP and NLP there are algorithms with polynomial complexity [12].

3. Surveyed Approaches

This section describes representative mapping and scheduling methodologies for NoC and bus-based MPES.

3.1 Approaches for NoC

3.1.1 Approaches without Network Assignment

This approach was developed by G. Varatkar et al. [2] and consists in a two-step methodology for minimizing the overall energy consumption in NoC. The communication-aware step performs simultaneous mapping and scheduling of tasks aiming to reduce the communication energy consumption by minimizing the inter-processor communication volume. It also facilitates task energy minimization in the voltage selection step, by maximizing the slack. The two optimization goals are alternated by a communication criterion, which has the role of keeping the local inter-processor communication volume of incoming edges under a limit stated globally which depends on the application communication volume and a factor K ($0 \leq K \leq 10$).

T. Lei et al. [4], [5] use a two-step GA for task mapping and then applies ASAP/ALAP techniques for task scheduling. The mapping goal is to maximize the timing performances, while scheduling is employed to check the hard deadlines. The communication is not mapped and scheduled and its delay is estimated using the average distance in NoC or the Manhattan distance between processors.

An energy-aware methodology for NoC was developed by J. Hu et al. [6] which performs communication mapping and scheduling to minimum available path and uses overall energy-consumption-gap to decide between possible task mappings and schedulings when building an initial solution.

3.1.2 Approaches with Network Assignment

D. Shin et al. [7] have proposed a methodology with network assignment and link speed allocation for reducing communication energy in NoC with voltage scalable links, while verifying the hard real time constraints. Task mapping verifies also area constraints. Hard deadlines are guaranteed, because even if communication scheduling is not performed, the worst case communication delay of links is used. The methodology uses GA for task mapping and network assignment and LS for task scheduling and link speed assignment.

GA for task mapping uses a mapping chromosome, a two-point crossover and random mutation. GA for tile allocation uses permutations of tiles as chromosome, cycle crossover to generate only legal solutions, and random exchanges as mutation. GA for routing path allocation uses a binary chromosome to encode moves along the X direction and Y direction, a coordinate crossover with crossover point at intersection of paths and a random mutation operator which exchange the locus of opposite directions. GA for routing path allocation has an impact on communication volume of each link and thus on link delay.

All approaches aim to meet hard deadlines, but only two guarantee them [6], [7]. The methodology of T. Lei et al. [4] maximizes performances for a set of applications. None of the approaches deal with soft deadlines. Area constraints are verified only by D. Shin et al. [7]. Simultaneous mapping and scheduling is sometimes performed to obtain a higher quality solution. Two approaches [2], [6] use LS for simultaneous task mapping and scheduling, and another one [6] has developed a deterministic method for simultaneous communication mapping and scheduling.

4. Our approach

In the survey of papers looked before we have seen that all of these works don't take in the same time some important objectives such optimal mapping of tasks, communications and load balancing. For this we propose an approach which target mapping with Network Assignment for heterogeneous distributed embedded systems. The methodology has two nested loops: the outer loop for core allocation and task mapping and the inner loop for communication and assignment mapping (Figure 3).

The task mapping is treated separately from communication mapping and is implemented with a GA. Penalty factors are used such as area constraints, load balancing and memory usage. The communication mapping is carried on together with scheduling to better optimize the common goals. For this we use Dijkstra's method to find the optimal path between two processors.

Both GA use integer chromosomes (priority/mapping strings), multi-objective fitness, single or two-point crossovers, random mutations with dynamic probability scheme (exponentially decreasing), ranking selection for parents and offsprings, random initial population, and population diversity as termination criterion.

4.1 Architectural Model and Application Model

The application model is a graph GT(T,M). T is a set of nodes representing tasks and M a set of edges representing communications. Several concurrent applications could be

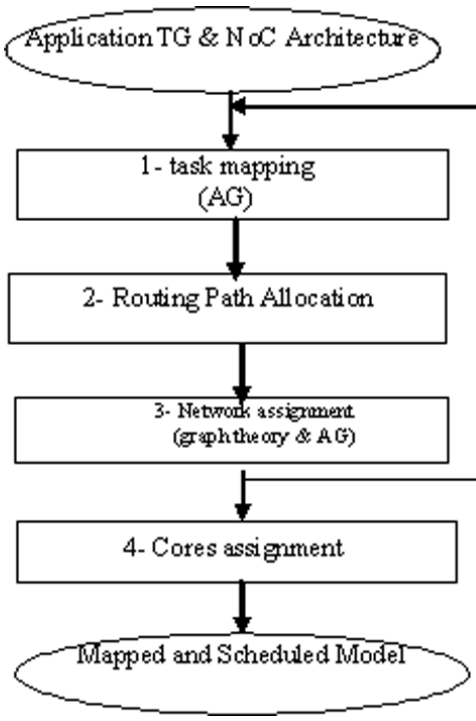

Application TG & NoC Architecture
1- task mapping (AG)
2- Routing Path Allocation
3- Network assignment (graph theory & AG)
4- Cores assignment
Mapped and Scheduled Model

Figure 3.

represented as a set of GT. Each node ti∈T (i=1,2,…, n) represents a task which sends and receives messages, an edge (t_i, t_j) describes the communication between two tasks t_i and t_j. ta_{ij} is the volume of communication between tasks (t_i, t_j), and bt_{ij} is the minimal bandwidth required for communicating tasks t_i and t_j.

The architectural model is also a graph GA= (S, B). An NoC is basically an heterogeneous multiprocessor system, PEs could be general purpose or special purpose processors, ASICs, FPGAs or memories of various types. Each node sq∈S (q=1.. m) represents a processor and each edge $b_{r\in}$ B (r=1,2,…, l) is a link between two nodes directly connected. There are few parameters which enclose the representation: bandwidth b_{ij} associated to each edge, speed (frequency), memory size, area constraints of PE, interconnection topology of PE.

4.2. Methodology

4.2.1 Optimization goals

The variable $X_{ij} \in \{0,1\}$ represent the affectation or not of the task t_i to the node s_j. Te_{ij} is the execution cost of task T_i on node s_j. The global execution cost for one node Sj is done in (1) :

$$\sum Te_{ij} X_{ij} \qquad i=1..n \qquad (1)$$

Te_{ij} is the execution time for task i on processor j. Global cost ($Cost_1$) for all nodes is the sum of (1) j=1 to m.

The communication global cost for a node S_k is:

$$\sum_{i=1}^{N} \sum_{j=i+1}^{N} C_{ijkl} X_{ik} X_{jl} \qquad l=1..m \qquad (2)$$

Then the goal is to reduce $C = (1)+(2)$ for $k = 1$ to m with $k \neq l$.

With AG algorithm we have fitness= C_{MAX} - C when C_{MAX} is the biggest cost then we can have and C the cost of one chromosome.

$$\text{Fitness} = \begin{cases} C_{Max} - C & \text{If constraints are verified} \\ C_{Max} - C * Pequi & \text{If load balancing is not verified} \\ C_{Max} - C * Pmem & \text{If memory is not verified} \\ C_{Max} - C * Pfil & \text{if buffer size not verified} \\ 0 \quad \text{if } C < C * Pénalité & \end{cases}$$

Then searching the good cost Someconstraints must be verified such:

Pequi is a penalty for load balancing, Pmem for memory and Pfil a penalty for buffer.

During search optimal path we use Dijkstra's algorithm to find the shortest distance between two PEs. We verify after if this path can support the assigned communication

4.2.2 Implementation and results

This work was realised under Windows environment with C++ Builder as tool language. For interface we have used graphical input for problems of small size and matricial input for problems of large size (figure4).

For our experiments we have used a same architecture with 9 PE and same topology and links. We have searched a good mapping for applications of different sizes (form 10 to 80 tasks) sharing the same topology and characteristics.

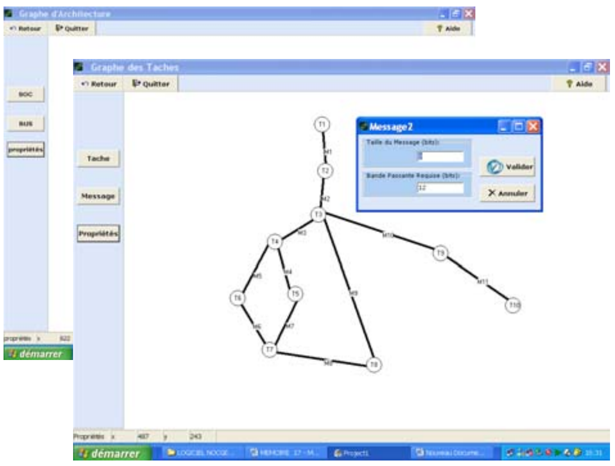
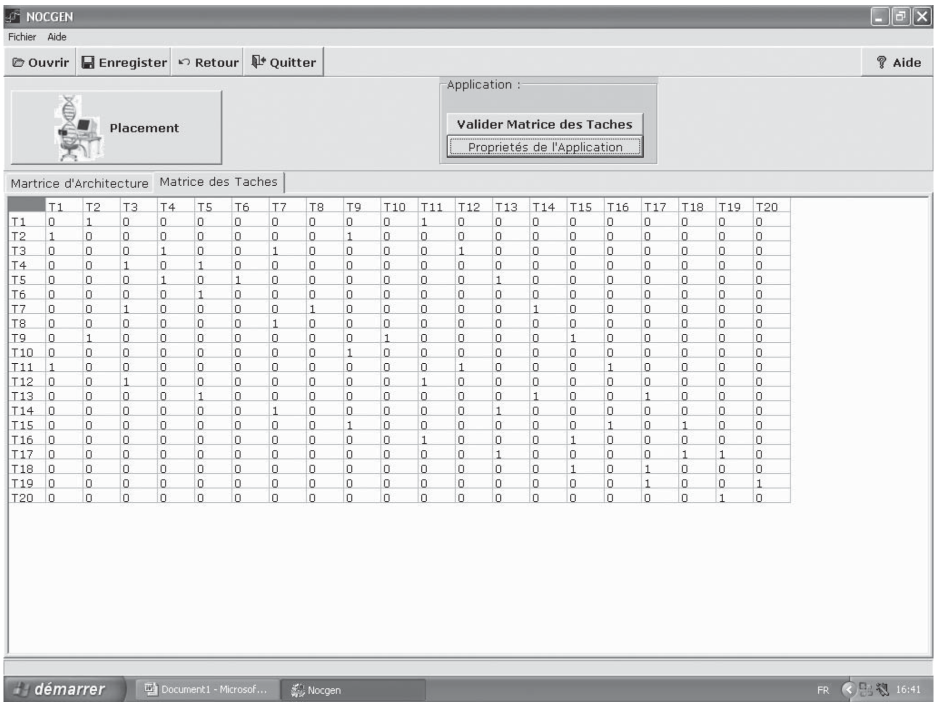


Figure 4. Matricial input for big size problem should be centralised

The results of different experiments for applications of different size that we have obtained are included in the flowing table.



	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	T15	T16	T17	T18	T19	T20
T1	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
T2	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
T3	0	0	0	1	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0
T4	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
T5	0	0	0	1	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0
T6	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
T7	0	0	1	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0
T8	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
T9	0	1	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0
T10	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
T11	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0
T12	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
T13	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0
T14	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0
T15	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	1	0	0
T16	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0
T17	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	1	0
T18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0
T19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1
T20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

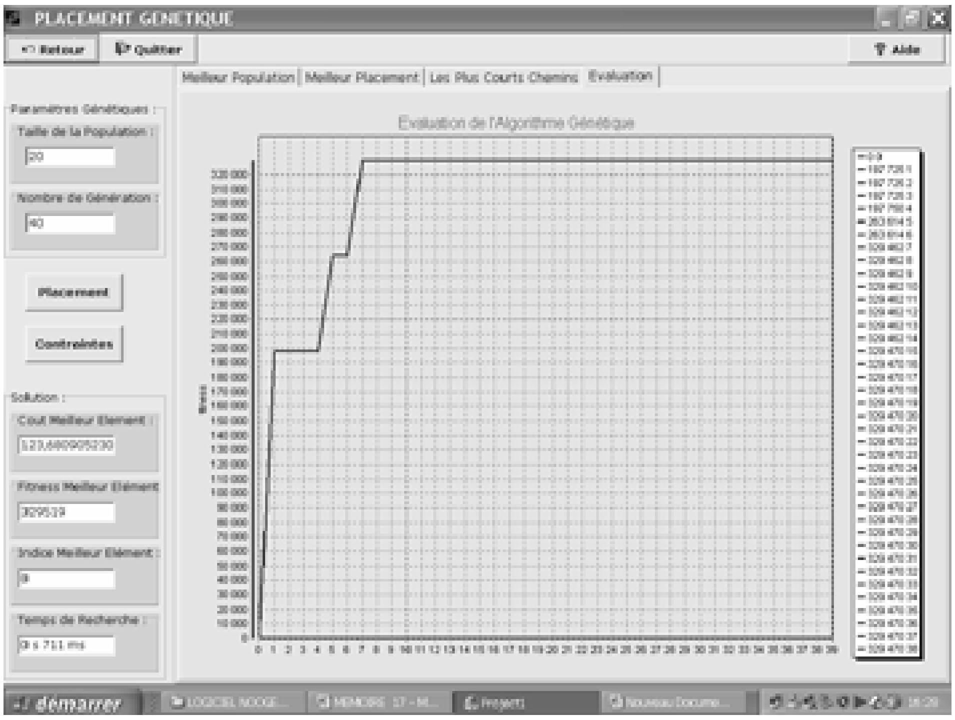
Figure 5. Graphical interfaces for input NoC and Application

Size	Fitness	Time execution
10	188290	0s440
20	376614	0s741
30	470576	0s961
40	611731	1s281
50	517634	1s32
60	517670	1s51
70	564774	1s732
80	517770	1s833

The achieved results and performances (figure 5 & 6) indicate that the proposed algorithm has a polynomial complexity and that it is adapted for computationally hard problems.

5. Comparaison with branch and bound

In the goal to have an idea on the quality and time research of our approach we have implemented branch and bound as a exact method. Optimization of communication is found in



Paramètres Génétiques :
Taille de la Population : 20
Nombre de Génération : 40

Solution :
Cout Meilleur Element : 123,680905230
Fitness Meilleur Element : 329519
Indice Meilleur Element : 8
Temps de Recherche : 0 s 711 ms

Figure 6.

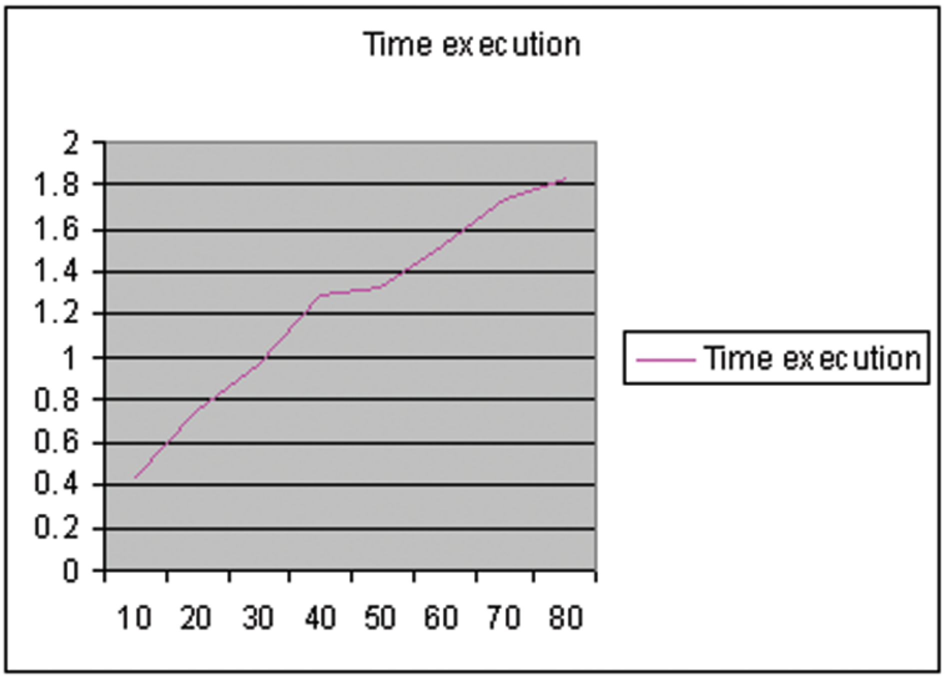


Figure 7.

same way and we have used dijkstra for the short path. The results of branch and bound are close to the results given by our method. But the time of research grow quickly when the size of problem increase.

The folowing table gives time execution or time research of each method and we can see that our methode converge quickly than the other with a nearly mapping.

Size	Methode based on Ga	Branch and bound method
10	0s440	0s600
20	0s741	1s986
30	0s961	3s550
40	1s281	5s230
50	1s32	9s320
60	1s51	13s213

6. Conclusion

In this paper an algorithm for allocation and scheduling has been proposed exploiting the method of AG and Disjkstra's shortest path algorithm. Under that we can't assert that this algorithm gives exact results for all kinds of problems. The proposed framework searches a good mapping for heterogeneous distributed embedded systems and of course this approach can easily be applied to regular architectures and first experimental results show that the global time for research is reasonable. Nevertheless for asserting that this method can be used for solving problems with large sizes we must experiment the algorithm using examples with many hundreds of PE and tasks. That will be the continuation for this work and our perspective.

References

[1] Pop, Ruxandra., Kumar, S. A Survey of Techniques for Maping and Scheduling Application to Network on Chip Systems, Research Report 04:4

[2] Varatkar, G., Marculescu, R (2003). Communication -Aware task Scheduling and Voltage Selection for Total Systems Energy Minimization, in Proc. IEEE/ACM Intl.Conf. On Computer Aided Design, San Jose, CA, Nov.

[3] Zhang, Y., Hu, X., Chen, D (2002). Task Scheduling and Voltage Selcetion For Energie Minimisation, DAC.

[4] Lei, T., Kumar, S (2003). A Two-step Genetic Algorithme for Mapping Task Graphs to a NoC Architecture.

[5] Lei, T., Kumar, S (2003). Algorithms and Tools for Noc Based System design, SBCCI 2003.

[6] Hu, J., Marculescu, R (2004).Energy-Aware Communication and Task Scheduling for Network-on-Chip Architecture under Real-Time Constraints", *In*: Proc. Design, Automation and Test in Europe Conf., Paris, France, Feb.

[7] Shin, D., Kim, J (2004). Power-Aware Communication Optimisation for Networks-on-Chips with Voltage Scalable Links, *In*: Proc. CODES+ISSS'04, p.170-175, Sep.

[8] Andrei, M.T., Schmitz, P., Eles, Peng, Z., Al-Hashimi, B.M (2004). Overhead-Conscious Voltage Selection for Dynamic and Leakage Power reduction of Time-Constraint Systems, *In:* Proc. Design, Automation and Test Europe Conference (DATE2004) Paris.

[9] Heath, M (2004). Synchro-Tokens: Eliminating Nondeterminism to Enable Chip-Level Test of Globally Asynchronous Locally Synchronous SoC's, Proceedings.

[10] Lyonnard, D (2003). *Approche d'assemblage*

systématique d'éléments d'interface pour la génération d'architecture *multiprocesseur* , Thèse de Doctorat, Institut National Polytechnique de Grenoble. Directeur de Thèse: A. A. Jerraya.

[11] Mazzour, E. et Hordouin, D. *Une aide algorithmique à l'optimisation du placement des capteurs dans un procédé"*, Département de génie des mines, de la métallurgie et de matériaux, Université Laval, Québec. Décembre.

[12] L. Benini, L., Bertozzi, D., Guerri, A., Milano, M (2005). Allocation and scheduling for MPSoCs via decomposition and no-good generation, *In*: Proc. of Intern. Joint Conf. on Artificial Intelligence (IJCAI05), p. 1517–1518.

[13] Luca Benini, Davide Bertozzi, Alessio Guerri, Michela Milano, "Allocation, Scheduling and Voltage Scaling on Energy Aware MPSoCs*",* CPAIOR 2006: 44-58