International Journal of Computational Intelligence Research.
ISSN 0973-1873 Vol.1, No.2 (2005), pp. 109–126

Mapping Cores on Network-on-Chip

Giuseppe Ascia, Vincenzo Catania and Maurizio Palesi

DIIT - University of Catania, Italy
{*gascia, vcatania, mpalesi*}*@diit.unict.it*

Abstract: **The paper addresses the problem of topological mapping of intellectual properties (IPs) on the tiles of a mesh-based network on chip (NoC) architecture. The aim is to obtain the Pareto mappings that maximize performance and minimize the amount of power consumption. As the problem is an NP-hard one, we propose a heuristic technique based on evolutionary computing to obtain an optimal approximation of the Pareto-optimal front in an efficient and accurate way. At the same time, two of the most widely-known approaches to mapping in mesh-based NoC architectures are extended in order to explore the mapping space in a multi-criteria mode. The approaches are then evaluated and compared, in terms of both accuracy and efficiency, on a platform based on an event-driven trace-based simulator which makes it possible to take account of important dynamic effects that have a great impact on mapping. The evaluation performed on real applications (an MPEG-4 codec and a cellular phone application) confirms the efficiency, accuracy and scalability of the proposed approach.**

I. Introduction

Continuous improvements in semiconductor technology mean that a whole processing system comprising processors, memories, accelerators, peripherals, etc. can now be integrated in a single silicon die. In addition, a reduction in the time-to-market has led researchers to define methods based on the reuse of pre-designed, pre-tested modules in the form of intellectual properties (IPs). Despite this, hardware designers are not yet able to fully exploit the abundance of transistors that can be integrated with current technology. Designer productivity, in fact, is growing by just 20% a year, as compared to an increase of over 60% a year by technology [33]. This gap will have to be reduced in order to respond to future requests by the consumer applications market (smart phones, automotive electronics, home networks, entertainment systems, etc.). Possible solutions to these problems can be sought in platform based design (PBD), which is based on the reuse of components, architectures, applications and implementations [8, 21, 26]. Of course the aim is always to obtain a good trade-off between generality and performance. Generality makes it possible to reuse hardware, software, development flows, etc., while performance (latency, cost, power, etc.) can be guaranteed by using specific dedicated architectures.

Without doubt, today, the on-chip interconnection system represents one of the major elements which has to be optimized in designing a complex digital system. The International Technology Roadmap for Semiconductors [33] foresees it will represents the limiting factor for performance and power consumption in next generation systems-on-a-chip (SOCs). The continuous reduction in the time-to-market required by the telecommunications, multimedia and consumer electronics market makes full-custom design of an interconnection system inappropriate and has led to the definition of design methodologies focusing on design reuse. This is confirmed by the great standardization effort made by the VSI Alliance [42] and the development, by the major EDA and Semiconductor companies, of on-chip interconnection systems that are easy to integrate and scale [20, 3, 29, 35, 36]. Although, however, they are good solutions for current SOCs integrating fewer than 5 processors and rarely more than 10 master buses, their use in next-generation systems, which are likely to integrate hundreds of modules, seems hardly feasible.

The limiting factor is mainly the topological organization of the interconnection between the various units, which will substantially remain bus-based. As regards performance, the continuous reduction in gate delays and increase in wiring delays will cause significant synchronization problems. In 50 nm technology, the projected chip die edge will be around 22 mm, with a clock frequency of 10 GHz. An optimistic estimate of the propagation delay for a signal crossing a chip diagonally ranges between 6 and 10 clock cycles[38]. At any rate, Moore's law will remain valid for the next 10 years and single processors will not be able to use all the transistors on a chip. Synchronous regions will occupy an increasingly lower fraction of a chip [37] giving rise to locally synchronous, globally asynchronous solutions [16]. Applications will be modelled as a set of communicating tasks with different characteristics (e.g. control-dominated, data-dominated) and origins (reused from previous projects or acquired from third parties), which will make implementations extremely heterogeneous.

A. Network on Chip

A type of architecture which lays emphasis on modularity and is intrinsically oriented towards supporting such heterogeneous implementations is represented by Network-on-Chip (NoC) architectures [12]. These architectures loosen the bottleneck due to delays in signal propagation in deep-submicron technologies and provide a natural solution to the problem of core reuse by standardising on-chip communications. The NoC architectural topology most frequently referred to can be represented by an $n \times m$ mesh. Each tile of the mesh contains a resource and a switch. Each switch is connected to a resource and the four adjacent switches. A resource is generally any core: a processor, a memory, an FPGA, a specific hardware block or any other IP compatible with with the NoC interface specifications. More generally, a resource may be represented by a complex multi-master and multi-slave system using an interconnection network based on shared-bus. The design flow for an architecture of this kind involves several steps. First the application has to be split up into a set of concurrent communicating tasks. Then the IPs are selected from the IP portfolio and the tasks are assigned and scheduled. Finally, the IPs have to be mapped onto the mesh in such a way as to optimise the metrics of interest.

The last phase is currently assuming more and more interest in the scientific community [17, 28]. Actually, it has a strong impact on typical performance indexes to be optimized. Unfortunately, the mapping problem is an istance of constrained quadratic assignment problem which is known to be NP-hard [14]. The search space of the problem increases factorially with the system size. It is therefore of strategic importance to define methods to search for a mapping that will optimise the desired performance indexes (performance, power consumption, quality of service, etc.) with a good tradeoff between accuracy and efficiency. This represents the main focus of this paper. In addition, these strategies have to a multi-criteria exploration of the space of possible architectural mapping alternatives. The objectives to be optimised are, in fact, frequently multiple rather than single, and are almost always in contrast with each other. There is therefore no single solution to the problem of exploration (i.e. a single mapping) but a set of equivalent (i.e. not dominated) possible architectural alternatives, featuring a different trade-off between the values of the objectives to be optimised (Pareto-set).

B. Contribution

In this paper we present a multi-objective exploration approach for the mapping space of a mesh-based NoC architecture. The approach, based on evolutionary computing techniques, is an efficient and accurate way to obtain the Pareto mappings that optimize performance and power consumption. In addition, two of the most widely known approaches to topological mapping of IPs in a mesh-based NoC architecture [17, 28] have been extended to achieve multi-criteria optimization and have been compared with the approach proposed here. In contrast with the approaches in the existing literature which use static analysis to evaluate a mapping, here we use an event-driven trace-based simulator which makes it possible to take account of important dynamic effects that have a great impact on performance indexes to be optimise. To the best of our knowledge this work is the first attempt to attack the topological mapping problem for NoC architectures from a multi-objective point of view taking care of model important dynamic effect such as contention for outgoing links, backpressure effects, influence of buffer size, packet size, etc.

C. Paper Organization

The rest of the paper is organized as follows. Section II summarizes some of the most important contributions in the field of topological mapping of IPs/cores in mesh-based NoC architectures. Section III presents the simulation and evaluation framework used and the impact of the architectural and application parameters on the performance indexes considered. Section V our approach for exploration of the mapping space is presented. In the same section we discuss the multiobjective extension of two other algorithms proposed in literature we compare to. Experimental results are reported in Section VI. Finally, Section VII summarizes our contribution and outlines some directions for future work.

II. Previous Work

The problem of mapping in mesh-based NoC architectures has been addressed in three previous papers. Hu and Marculescu [17] present a branch and bound algorithm for mapping IPs/cores in a mesh-based NoC architecture that minimizes the total amount of power consumed in communications with the constraint of performance handled via bandwidth reservation. The same authors in [19] extend the approach to constructs a deadlock-free deterministic routing function such that the total communication energy is minimized. Murali and De Micheli [28] address the problem under the bandwidth constraint with the aim of minimizing communication delay by exploiting the possibility of splitting traffic among various paths. Lei and Kumar [25] present an approach that uses genetic algorithms to map an application, described as a parameterized task graph, on a mesh-based NoC architecture. The algorithm finds a mapping of the vertices of the task graph on the available cores so as to minimize the execution time.

These papers do not, however, solve certain important issues. The first relates to the mapping evaluation model used, which can be defined as “static”. The exploration algorithm decides which mapping to explore without taking important dynamic effects of the system into consideration. For example, failure to model the effects of bus contention causes components which communicate with each other more fre-

quently to be clustered, whereas it may be more effective to separate components whose traffic flows overlap in time so as to increase the degree of concurrency. In the above-mentioned works, in fact, the application to be mapped is described using task graphs, as in [25], or simple variations such as the core graph in [28] or the application characterization graph (APCG) in [17]. These formalisms do not, however, capture important dynamics of communication traffic. They hypothesize worst-case conditions, which leads to several mappings being discarded and thus a highly conservative exploration. The second problem relates to the optimization method used. It refers in all cases to a single performance index (power in [17], performance in [28, 25]). As we will see in the section devoted to experiments, optimization of one performance index may lead to unacceptable values for another performance index (e.g. high performance levels but unacceptable power consumption). We therefore think that the problem of mapping can be more usefully solved in a multi-objective environment, i.e. one in which there is no single solution but a set of mapping alternatives (which we will indicate as Pareto mapping), each featuring a different tradeoff between performance indexes, from which the designer (or decision maker) will choose the most suitable.

The contribution we intend to make in this paper is to propose a multi-objective approach to solving the problem of mapping IPs/cores in mesh-based NoC architectures. The approach will use evolutionary computing techniques to explore the mapping space with the goal to optimize performace and power consumption. The mappings visited during the exploration process will be evaluated using a trace-based approach which gives an excellent combination of accuracy and efficiency features.

III. Evaluation of a Mapping

Figure 1 shows the NoC topology we will refer to. It is a two-

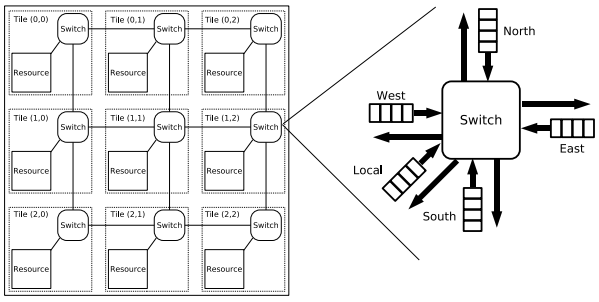


Figure. 1: Structure of a 3x3 mesh-based NoC architecture.

dimensional mesh of processing resources. Each processing resource is connected to the communication network by a switch. We will call the pair formed by a resource and a switch a *tile*. The term *mapping* will be used to indicate assignment of an IP/core to each tile in the NoC. Each switch in the NoC is connected to the four adjacent switches except for those at the network boundaries. Switches send data from one network interface to the other by means of packets. Such a packet consists of one or more *flow control digits* (or *flits*), were a flit is the minimal transmission unit. On each side of a switch there is an output and an input port. The input port has a finite-length FIFO buffer in which flits to be routed are queued. The use of the FIFO is regulated by back-pressure mechanism [18]. Under this scheme, a flit will be held in the buffer until the downstream router has empty space in the corresponding input FIFO. Thus, the network will not drop any packet in transit. This is extremely important for NoC architectures which may not implement very advanced end-to-end protocol.

The routing algorithm features *static XY* routing in which a flit is first routed in a horizontal direction (X) and then, when it reaches the column where the destination tile is located, it is routed in a vertical direction (Y). Of course the XY routing is a *minimal* path routing algorithm and is *free* of deadlock and livelock [15]. As a transmission scheme we use wormhole routing because of the low cost (the buffer capacity can be less than the length of a packet) and low latency (the router can start forwarding the first flit of a packet without waiting for the tail).

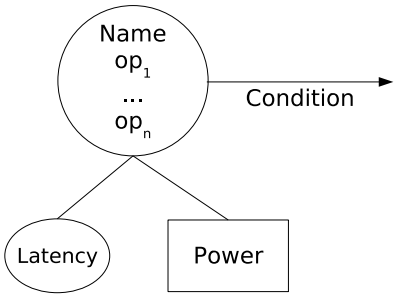


Figure. 2: Behavioural annotated graph (BAG).

To describe the functioning of the various components of the simulation framework we will use a representation based on a variation of a finite-state machine which we will indicate as a *behavioral annotated graph* (BAG). Each machine state is identified by a name, a set of operations ($op_1, \ldots, op_n$) and two attributes which we will call *latency* and *power* (See Figure 2). Transition from one state to another is represented by an oriented arc associated with a condition (transition only occurs when the condition is met). The conditions are evaluated after a time equal to the value of the attribute *latency*, starting from the instant at which the state is entered. If none of the conditions on the arcs are met, the machine remains in the current state and the process is repeated. Otherwise there is a state transition and the total amount of energy consumed while the machine remains in this state is measured. This can be summed up as follows:

1. $t_{enter} = \text{GetCurrentTime}()$;
2. $op_1, \ldots, op_n$;

3. if (GetCurrentTime() $-t_{enter} < latency$) goto 3;
4. if (Condition == false) goto 2;
5. $energy = (\text{GetCurrentTime}() - t_{enter}) \times power$;
6. State transition;

Figure 3 shows the BAG for a generic core. In the *Idle* state the average amount of power consumed by a core is $P_{idle}^{(core)}$. If there is at least one flit in the input queue it passes to a fictitious state (featuring $latency = 0$ and $power = 0$) in which the type of flit is evaluated. If it is the first in a data flow directed towards the core involved (head flit, *H*) or an intermediate flit (body flit, *B*) the core switches to the *Buffering* state, with an average power consumption of $P_{buff}^{(core)}$ and a latency of $T_{buff}^{(core)}$. This state allows us to simulate situations in which a core starts to process the data, not on a flit basis but on a set of data that cannot be contained in a single flit. If, on the other hand, the flit is the last in a communication flow, the core switches to the *Process* state in which the flit is actually processed, with an average power consumption of $P_{process}^{(core)}$ and a latency of $T_{process}^{(core)}$. The operation performed in both these states is to consume the flit at the head of the queue and then, when the latency time ends, to switch unconditionally back to the *Idle* state.

Figure 4(a) shows the interface of a switch. Each of the five input ports has an associated queue (buffer). Each output port is associated with an input signal (with the suffix *Ready*) which is asserted whenever the element connected to the relative port is ready to accept a flit. Figure 4(b) shows the BAG for a generic switch. In the *Idle* state a switch consumes on average $P_{idle}^{(switch)}$. If there is at least one flit in at least one of the 5 input queues the switch passes to a fictitious state (with $latency = 0$ and $power = 0$) in which the flit is read and immediately afterwards to the *Routing* state. In this state (which features an average power consumption of $P_{routing}^{(switch)}$ and a latency of $T_{routing}^{(switch)}$) the output port on which to route the flit is determined and the relative fictitious state (*LOCAL, NORTH, SOUTH, EAST, WEST*) is entered. Only when the flit is ready to be transmitted (*ready=true*) does the switch pass to the *Transmit* state in which the flit at the head of the input queue is extracted and then, on expiry of the latency time $T_{transmit}^{(switch)}$, unconditionally returns to the *Idle* state, with an average power consumption of $P_{transmit}^{(switch)}$, which models the power consumed on the interconnection buses between the switches.

The simulation is event-based and is performed by stimulating the network with concurrent trace files. Each trace file is a sequential list of communication patterns. Each pattern comprises three fields: a source identifier, a destination identifier, and the amount of information exchanged. The amount of traffic sent by the source core to the destination core is subdivided into packets and each packet is routed according to the routing scheme and BAGs described above.

A. Motivation

In this section we wish to demonstrate (using an experiment-based approach) that accurate modeling of the communication dynamics is essential in order to evaluate a network.

We will begin out analysis by considering as our performance parameter the speed at which a network handles a certain amount of incoming traffic. This mainly depends on the speed at which the switches route packets. If, for example a switch *A* has to forward the packet at the head of the input queue from its port α to the port β in the adjacent switch *B*, two events can occur: (i) the input queue in the port β is not full, or (ii) the input queue in the port β is full. In the former case, *A* can forward the packet, thus freeing a slot in the queue in port α. In the latter case, *A* has to wait for *B* to eliminate at least one packet from the input queue in port β before it can forward the packet. In general, therefore, the overall performance of the network (measured as the time required to handle all the incoming traffic) improves if the size of the switch input queues increases. With an increased input queue capacity, in fact, a generic switch needing to forward a packet to another switch will have a greater probability of being able to queue the packet in the input port of the other switch.

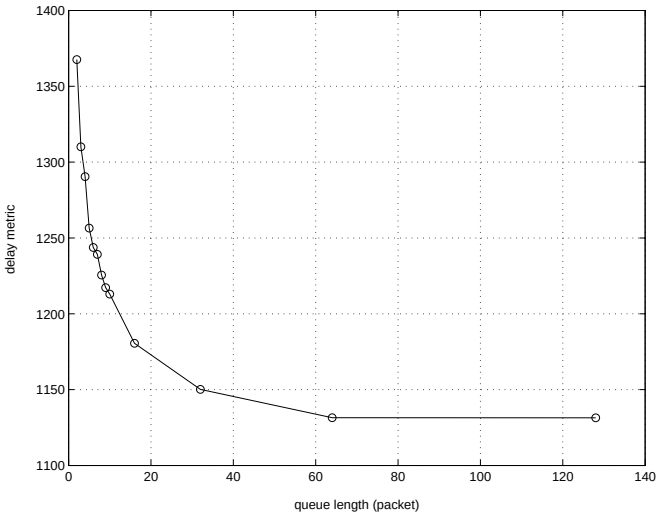


Figure. 5: Traffic draining time vs. switch input queue size.

Figure 5 shows the time required to handle traffic versus the size of input queues in the switch ports. The values were obtained on a 5x5 network. The latency and power attributes of the core BAGs were randomly set between 0 and 1 for each core and 0.1 for all the switches. The traffic was generated considering communication between the network nodes to be equally probable (that is, the probability that node *A* will communicate with node *B* is equal to the probability that node *C* will communicate with node *D*, however *A*, *B*, *C* and *D* are taken). The flow of data exchanged between two nodes has a Gausssian distribution with an average of 128 bytes and a variance of 64 bytes. Eight different traces formed by 100

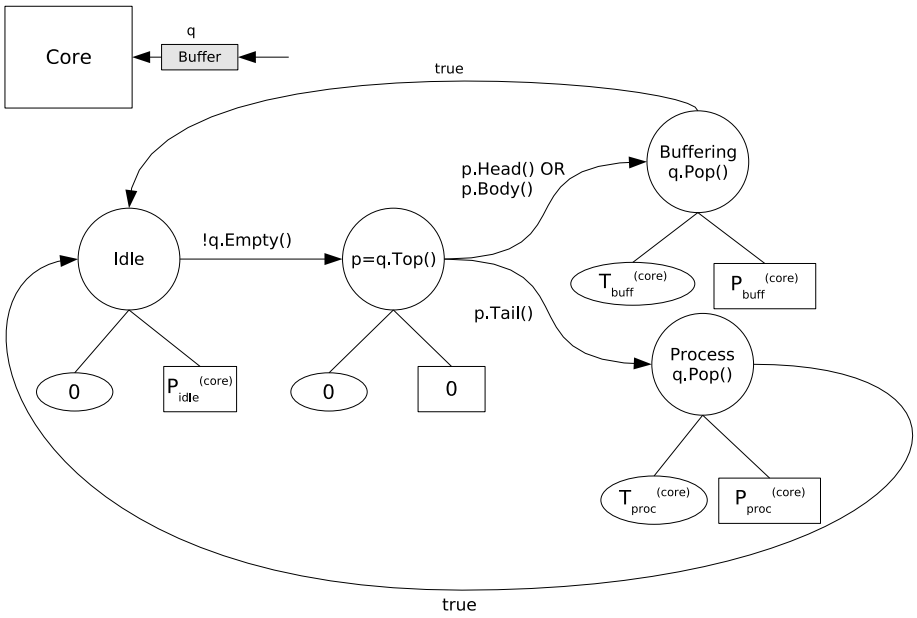


Figure. 3: Behavioural annotated graph of a generic core.

patterns were injected in parallel, so as to simulate 8 concurrent communications at each instant. Each point in the graph was obtained by measuring the time taken to handle the traffic in 100 different mappings and calculating the average value.

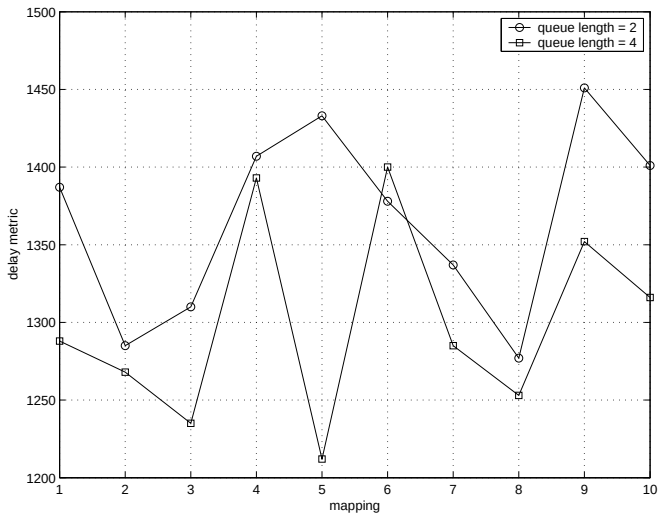


Figure. 6: Traffic draining time for 10 different mappings and two different networks (with switch input queues of 2 and 4 packets).

It can, however, be observed that in some cases an increase in the size of the switch queues may increase the traffic handling time. Figure 6 shows this possibility. It gives the traffic handling time for 10 different mappings with switches having input queues that allow a maximum of two and four packets to be queued. The traffic handling times for the second network are generally shorter than those for the first network, with one exception. With mapping 6, in fact, the traffic is handled faster in the first network. This behavior can only be detected via a dynamic analysis of the system, that is by taking into account the dynamic interaction between the various traffic flows, which is only possible by performing trace-based simulations.

It should also be observed that the optimal mapping is greatly affected by the architectural parameters of the network. Let us consider, for example, the size of the switch input buffers. In Figure 6 it can be seen that a mapping may be optimal for one network but not for another. Of the 10 mappings considered, in fact, mapping 5 is by far the best for the second network but the second worst for the first network.

To evaluate the impact of mapping and relate it to the traffic characteristics the following experiment was performed. 1000 mappings were randomly generated for each network $n \times n, n \in \{3,4,5\}$. $\lceil n^2/2 \rceil$ simulations were run for each mapping, relating to different traffic scenarios. These scenarios differed in the number of pairs of cores simultaneously communicating with each other. They range from an absolute lack of concurrency (that is, one and only one pair of cores are communicating at any one time) to maximum concurrency (at any one time there are $\lceil n^2/2 \rceil$ pairs of cores communicating with each other). Figure 7 shows the relationship between the maximum and minimum traffic draining times for 1,000 random mappings in the traffic scenarios described above. As can be seen, when the size of the network increases, so does the impact of mapping on performance. For a 5x5 network, for example, choosing a suitable mapping can improve performance by over 40%. It should be pointed

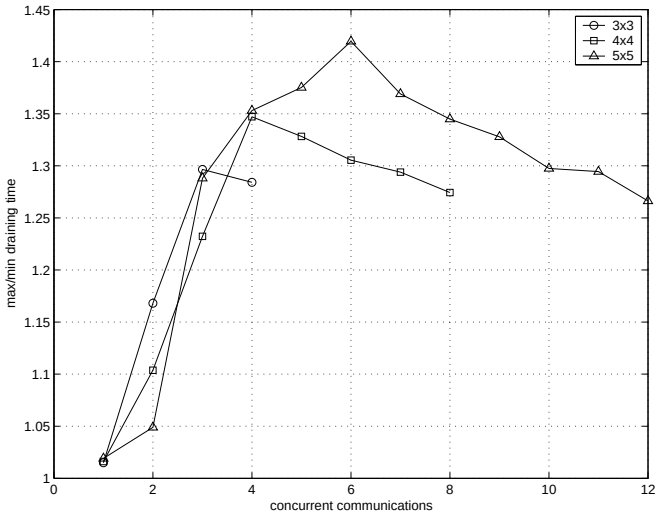


Figure. 7: Relation between maximum and minimum traffic draining time for 1,000 random mappings with varying numbers of concurrent communications and different network sizes.

out that these values are extremely conservative. They were obtained considering only 1,000 random mappings as compared with the $25! \simeq 10^{25}$ that are possible. It should also be noted that the impact of mapping depends greatly on the traffic characteristics. In all the cases considered, the maximum impact is obtained in traffic scenarios in which the number of pairs of cores communicating concurrently is equal to half the maximum number of pairs that can communicate concurrently.

IV. Exploration Framework

Figure 8 shows the framework for exploration of the space of possible mappings in mesh-based NoC architectures.

It comprises two macro blocks: a *NoC simulator* (to evaluate the performance indexes to be optimized for any mapping), and an *Exploration engine* (which determines the next mapping to be evaluated). The inputs to the framework are:

- *Architectural parameters*: for example, topology, network size, communication protocols, size of buffers in switches, priority assignment schemes, etc.

- *Application parameters*: these mainly refer to the characteristics of the communication traffic involved in the application being considered. They may relate to both the characterization of statistical models of the traffic exchanged between the various network resources, and real traces obtained by measuring the communication traffic during execution of the application. Useful application parameters to specify traffic in statistical models are: packet generation rate (packets can be generated at random or periodical intervals, or in a bursty or uniform fashion); the statistical distribution of the destination addresses (random, or polarized towards a certain group of resources (hot spot) etc.). For the real traces, they can be obtained from the communication task graph in which the application tasks are assigned and scheduled with reference to a library of available IPs/cores.

- *Set of BAGs*: these specify the functional behavior of each element in the NoC and also contain characterization information for estimation of the timing and power consumption parameters.

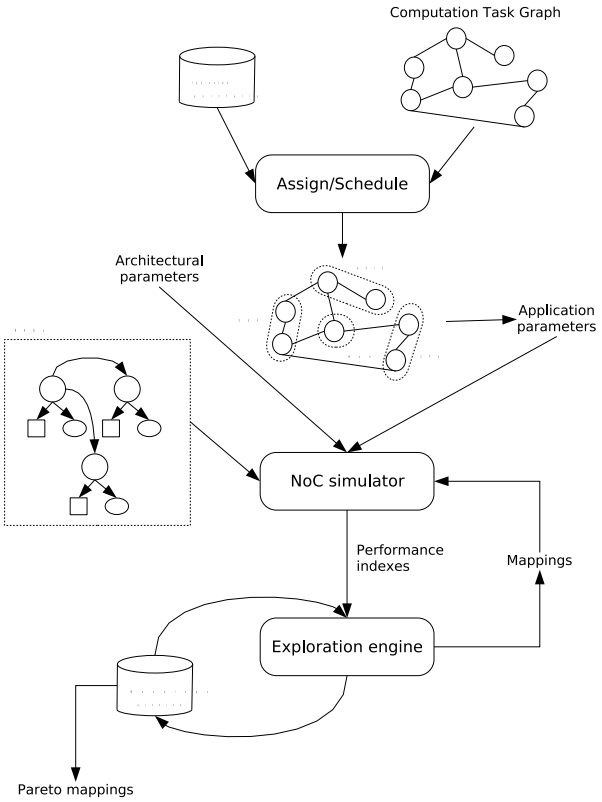


Figure. 8: Framework for simulation and exploration of the mapping space.

The flow of operations involved in exploration generally consists of repeating two phases: evaluation of one or more mapping alternatives, and determination of the next mapping/s to be evaluated. The first phase is carried out using a NoC simulator, which evaluates the performance indexes to be optimized. These represent the input for the second phase, which implements the exploration algorithm and produces the next mapping/s to be evaluated. The mappings evaluated are stored and can be used by the exploration algorithm to decide the next step. This iterative process is concluded when a stop criterion is met. Then the non-dominated mappings (Pareto mappings) are extracted from the mappings evaluated.

In this paper we will focus on the second phase of the framework, the one referring to the mapping space explo-

ration algorithms.

V. Multi-Objective Exploration of the Mapping Space

The mapping problem is an instance of a constrained quadratic assignment problem which is known to be *NP-hard* [14]. The search for an optimal mapping (henceforward referred to as exploration) is also complicated when the concept of optimality is not limited to a single performance index (or objective) but comprises several contrasting indexes. The traditional approach to a multi-objective optimization is to aggregate the objectives into a single one by means of a *weighting mean*. The main drawback to this approach is that it does not cover the non-convex regions of the Pareto-front and requires several instances of the optimization algorithm to be run with different weights. In this section we present: 1) an approach to multi-objective mapping space exploration that uses evolutionary algorithms as the optimization strategy; 2) multi-objective extension of an exploration algorithm based on the branch-and-bound proposed in [17]; and 3) multi-objective extension of a variation of the exploration algorithm proposed in [28].

A. Problem Formulation

The mapping problem can be expressed by Figure 9. Given a target application described as a set of concurrent tasks which have been assigned and scheduled, to exploit such an architecture, the fundamental questions to answer are: i) which tile each IP should be mapped to, ii) what routing algorithm is suitable for directing the information among tiles, such that the metrics of interest are optimized. More precisely, in order to get the best power/performance tradeoff, the designer needs to determine the topological place-ment of these IPs onto different tiles. Referring to Figure 9, this means to determine, for instance, onto which tile [e.g. (3,1), (1,3) etc.] each IP (e.g. DSP2, ASIC1 etc.) should be placed. While task assignment and scheduling problems have been addressed before [9], the mapping and routing problems described above represent a new challenge, especially in the context of the regular tile-based architecture, as this significantly impacts the energy and performance metrics of the system.

Formally, if C is the set of cores, and T the set of tiles, we will use the term *mapping* to indicate an injective and surjective function $M : C \rightarrow T$ that associates the tile $t \in T$ on which c is mapped with each $c \in C$.

Evaluating a mapping means obtaining the related performance indexes for a specific traffic scenario. If S indicates a traffic scenario, we define the *evaluation function*

$$\mathbf{V}(S,M) = (V_1(S,M), V_2(S,M), \ldots, V_n(S,M)) \qquad (1)$$

which yields the values of the n performance indexes relating to the mapping M for the traffic scenario S. In our case study,

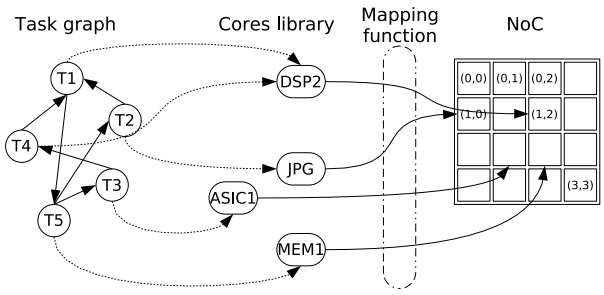


Figure. 9: Graphic explaination of the mapping problem.

for example, the evaluation function corresponds to the simulation framework (described in [5]) and the performance indexes are those the platform is capable of measuring (power, communication latency, bandwidth, throughput, etc.). Evaluation of an incomplete mapping made up of a set of cores $C' \subset C$ with a traffic scenario S is performed by evaluating the mapping on a traffic S' obtained by filtering out all communication flows in which the source or destination is a core $c \in C'$.

Given a traffic scenario S and two mappings M_1 and M_2, M_1 can be said to *dominate* M_2 (which will be indicated as $M_1 \succ M_2$) if $V_i(S,M_1) \leq V_i(S,M_2)$, $i \in \{1,2,\ldots,n\}$ and there exists at least one $j \in \{1,2,\ldots,n\}$ such that $V_j(S,M_1) < V_j(S,M_2)$. The *set of Pareto mappings* is a set of mappings that do not dominate each other. The Pareto front is the image of the evaluation function for the set of Pareto mappings. If $\mathscr{M}$ is the set of all possible mappings, the Pareto-optimal set $\mathscr{P}$ is the set of Pareto mappings such that $\nexists\, M \in \mathscr{M} : M \succ M'$, $M' \in \mathscr{P}$.

The aim of the approach we propose is to obtain as accurate an approximation as possible of the Pareto-optimal front by evaluating (visiting) as few mappings as possible. The optimization metrics we consider are the completion time and the total energy consumption. Formally, given a set of cores C, a set of tiles T (such that $|C| = |T|$), and a communication traffic scenario S (which models the communication between the cores $c \in C$), the topological mapping problem is the problem of finding the Pareto-optimal set $\mathscr{P}$ of mappings that optimise in both completion time and total energy consumption.

B. GA-based Multi-Objective Exploration of the Mapping Space

The use of evolutionary algorithms (EAs) as a multi-objective optimization technique is of increasing appeal. The fields of application are numerous, including among others computer science, engineering, economics, finance, industry, physics, chemistry, and ecology. EAs have been demonstrated to be very powerful and generally applicable for solving difficult multi-objective problems. Such algorithms create an interesting alternative to other approaches since they can be scaled with the problem size and can be easily run on

parallel computer systems. In VLSI design, EAs have been applied to a very broad range of problems: in problems relating to layout such as partitioning [1], placement and routing [40], in design problems including power low-power synthesis [7], technology mapping [23] and netlist partitioning [2].

There are various approaches to GA-based multi-objective optimization, divided into three main categories [10]:

1. Approaches using aggregation functions,
2. Approaches not based on the notion of Pareto optimum,
3. Pareto-based approaches.

The first type (those that use aggregating functions) reduce the problem of multi-objective optimization to one of scalar optimization by aggregation of the objective functions [41, 32]. The main disadvantage to aggregation functions is that they do not generate proper Pareto-optimal solutions in the presence of non-convex search spaces, which is a serious drawback in most real-world applications.

The second approach (not based on the notion of Pareto optimum) solves some of the difficulties encountered in the first, such as finding suitable weights to combine (aggregate) the objectives. One of the most famous approaches in this class is VEGA [32]. The drawback to this approach is that even if the user defines the objective functions independently of each other, the algorithm combines them. Thus, under certain conditions (i.e., when proportional selection is used) the resulting fitness function turns out to be a linear combination of the objective functions in which the weights depend on the distribution of the population in each generation. The problems are thus the same as those of the first approach i.e. not finding certain points in concave regions.

Currently, the third class of approaches (Pareto-based approaches) is the most promising. The basic algorithm consists of selecting Pareto non-dominated individuals from the rest of the population. These individuals are then assigned the highest rank and eliminated from further contention. Another set of Pareto non-dominated individuals are determined from the remaining population and are assigned the next highest rank. The procedure is repeated until the whole population is suitably ranked. The most widely used approaches belonging to this class are NSGA-II [13], PESA [11] and SPEA2 [43]. A simple steady-state Pareto-based evolutionary algorithm is presented in [39] that uses an elitist strategy for replacement and simple uniform scheme for selection. Here, no fitness calculations, ranking, sub-populations, niches or auxiliary populations are required.

In this paper we propose the use of a heuristic technique based on EAs for multi-objective mapping space exploration. More specifically, we chose SPEA2, which is very effective in sampling from along the entire Pareto-optimal front and distributing the solutions generated over the trade-off surface. SPEA2 is an evolution of SPEA [44] and incorporates a fine-grained fitness assignment strategy, a density estimation technique, and an enhanced archive truncation method. The chromosome is a representation of the solution to the problem, which in this case is described by the mapping. Each tile in the mesh has an associated gene which encodes the identifier of the core mapped in the tile. In an $n \times m$ mesh, for example, the chromosome is formed by $n \times m$ genes. The i-th gene encodes the identifier of the core in the tile in row $\lceil i/n \rceil$ and column $i\%m$ (where the symbol % indicates the modulus operator).

The *crossover* and *mutation* genetic operators were have been suitably redefined. More specifically, a crossover between two mappings M_f and M_m generates two new mappings M_{s1} and M_{s2} constructed as follows. Let $t_{x,y} \in T$ be the tile in row y and column x. Given a mesh of H rows and W columns, two random numbers $x \in \{1,2,\ldots,W-R\}$ and $y \in \{1,2,\ldots,H-R\}$ (where R is a user defined parameter) are extracted. Then, the crossover operator simply swaps the two regions consisting of tiles from $t_{x,y}$ to $t_{x+R,y+R}$. Figure 10 describes the crossover operator. Where the func-

```
1  (Mapping, Mapping)
2  XOver(Mapping M_f, Mapping M_m)
3  {
4      Mapping M'_f ← M_f;
5      Mapping M'_m ← M_m;
6      x = Rand({1,2,...,W−R});
7      y = Rand({1,2,...,H−R});
8
9      for (x',y') ∈ {x,x+1,...,x+R−1} × {y,y+1,...,y+R−1}
10     {
11         Swap(M'_f, M_f^{-1}(t_{x',y'}), M_m^{-1}(t_{x',y'}));
12         Swap(M'_m, M_m^{-1}(t_{x',y'}), M_f^{-1}(t_{x',y'}));
13     }
14
15     return (M'_f, M'_m);
16 }
```

Figure. 10: Crossover operator.

tion `Swap(`M`,`c_1`,`c_2`)` exchanges the core c_1 with the core c_2 from mapping M.

The mutation operator acts on a single mapping M to obtain the mutated mapping M' as follows. A tile T_s from mapping M is chosen at random. Indicating the core in the tile T_s as c_s and c_t as the core with which c_s communicates most frequently, c_s is remapped on a tile adjacent to T_s so as to reduce the distance between c_s and c_t by a hop, thus obtaining the mutated mapping M'. Figure 11 describes the mutation operator. The `RandomTile(`M`)` function gives a tile chosen at random from mapping M. The `MaxCommunication(`c`)` function gives the core with which c communicates most frequently. If two or more cores have equal frequency of communication, one of them is randomly chosen. The `Row(`M`,`T`)` and `Col(`M`,`T`)` functions respectively give the row and column of the tile T in mapping M. Finally, the `North, South, East, West(`M`,`T`)`

```
1  Mapping Mutate(Mapping M)
2  {
3     Mapping M' = M;
4
5     Tile T_s = RandomTile(M');
6     Core c_s = M'^-1(T_s);
7     Core c_t = MaxCommunication(c_s);
8     Tile T_t = M'(c_t);
9
10    Tile T'_s;
11    if (Row(M', T_s) < Row(M', T_t))
12       T'_s = North(M', T_s);
13    else if (Row(M', T_s) > Row(M', T_t))
14       T'_s = South(M', T_s);
15    else if (Col(M', T_s) < Col(M', T_t))
16       T'_s = East(M', T_s);
17    else
18       T'_s = West(M', T_s);
19
20    Swap(M', T_s, T'_s);
21
22    return M';
23 }
```

Figure. 11: Mutation operator.

functions give the tile to the north, south, east and west of the tile T in mapping M.

C. Pareto-based Branch-and-Bound Approach

In [17] Hu and Marculescu present an approach using branch-and-bound as the mapping space exploration strategy. The approach is, however, a mono-objective one. In this subsection we will extend their approach in order to perform multi-objective exploration of the mapping space. We will call our approach *Pareto-based Branch-and-Bound* (*PBBB*).

Let $\{c_1, c_2, \ldots, c_{N^2}\}$ be the set of cores in the system in decreasing order with respect to the communication traffic. The core c_1 can be mapped on any of the N^2 tiles in the mesh. These N^2 mappings generate the first layer of a tree which is the starting point for the branch-and-bound algorithm. For each first-level mapping the core c_2 can be mapped on any of the N^2-1 free tiles, thus generating a second level $N^2 \times (N^2-1)$ mappings. This is the *branch* phase of the algorithm and is described in pseudo-code in Figure 12. Where the `MakeMappings(M,c)` function, given a mapping template M and a core c, yields a set of mappings obtained by mapping c on each free tile in M.

Each mapping at this level is evaluated (simulated) and then characterized according to the optimization objectives, which in our case are power and delay. The dominated mappings are discarded, while the branch and bound phases are reiterated on the survivors. This is the *bound* phase of the algorithm as described in pseudo-code in Figure 13. Where the `ExtractPareto(`$\mathscr{M}$`)` function extracts the

```
1  Mappings Branch(Mappings M, Core c)
2  {
3     Mappings M' = ∅;
4
5     for (M ∈ M)
6        M' = M' ⋃ MakeMappings(M, c);
7
8     return M';
9  }
```

Figure. 12: Branch phase of the branch-and-bound algorithm.

```
1  Mappings Bound(Mappings M)
2  {
3     Mappings M' = ExtractPareto(M);
4
5     if (|M'| > T_pbbb)
6        Pruning(M', T_pbbb);
7
8     return M';
9  }
```

Figure. 13: Bound phase of the branch-and-bound algorithm.

non-dominated mappings from the set $\mathscr{M}$. To prevent the algorithm from degenerating the bound phase is followed by a further pruning phase. Let us indicate the set of mappings generated by the bound phase as $\mathscr{M}$. If $|\mathscr{M}| > T$ (where T is a user-defined threshold) $|\mathscr{M}| - T$ mappings are eliminated at random from $\mathscr{M}$. The `Pruning(`$\mathscr{M}$`,`T_{pbbb}`)` function randomly eliminates mappings from a set $\mathscr{M}$ if the cardinality of this set exceeds a threshold T_{pbbb} in such a way as to make the cardinality of $\mathscr{M}$ equal to T_{pbbb}.

The branch and bound phases are reiterated until all the cores have been mapped. For example, indicating the mappings obtained in the bound phase as $M_1, M_2, \ldots, M_n$, the core c_3 will be mapped for each of them on to the N^2-2 possible tiles. The $n \times (N^2-2)$ mappings will be the third level of the tree. The algorithm terminates when all the cores have been mapped and the leaves of the tree will be the Pareto mappings. A pseudo-code description of *PBBB* is given in Figure 14. Where the `SortByTraffic(`C`)` function orders the set of cores C according to the communication traffic.

D. Pareto-based NMAP Approach

Murali and De Micheli in [28] propose NMAP, an algorithm that maps the cores in a mesh NoC architecture with the aim of minimizing the average communication delay. In this subsection we will extend NMAP to perform a multi-objective exploration of the mapping space. Unlike [28], however, we will refer to a routing XY. We will call this approach *Pareto-based NMAP* (*PBNMAP*).

```
Mappings PBBB(Cores C)
{
   Cores Cs = SortByTraffic(C);
   Mappings M = MakeMappings(∅, Cs,1);
   for (c ∈ Cs \ {Cs,1}) {
      M = Branch(M, c);
      M = Bound(M);
   }

   return M;
}
```

Figure. 14: Pareto-based branch-and-bound approach.

The algorithm comprises two phases. In the first the cores featuring the largest amount of communication traffic are mapped onto the central tiles in the mesh (i.e. the $(N-2)\times(N-2)$ tiles with the greatest numbers of neighbours). The remaining cores are then ordered in decreasing order with respect to the communication traffic they have with the cores mapped in the previous phase. The first, c_1, is mapped onto each of the $4\times(N-1)$ remaining tiles. The $4\times(N-1)$ are evaluated and those that are dominated are discarded. If $\mathscr{M}_1$ is the set of non-dominated mappings, the algorithm is reiterated for each $M\in\mathscr{M}_1$ with c_2 and so on until the last core $c_{4(N-1)}$ has been mapped and the set of Pareto mappings $\mathscr{M}=\mathscr{M}_{4(N-1)}$ has been obtained. Figure 15 gives the pseudo-code for this first phase. Where

```
Mappings PBNMAP_1st(Cores C)
{
   Cores Cs = SortByTraffic(C)
   Mapping M;
   for (i ∈ {1,2,...,(N-2)×(N-2)})
      Map(M, Cs,i, (i-1)/(N-2)+1,
          (i-1)%(N-2)+1);

   Cores C2s =
      SortByC2CTraffic({Cs,(N-2)*(N-2)+1,...,Cs,N²},
                       {Cs,1,...,Cs,(N-2)*(N-2)});
   Mappings M = ∅;
   for (c ∈ C2s) {
      M = MakeMappings(M, c);
      M = ExtractPareto(M);
   }

   return M;
}
```

Figure. 15: First phase of the Pareto-based NMAP approach.

the `Map(`M,c,row,col`)` function maps core c onto the tile in row *row* and column *col* of the mapping M. The `SortByC2CTraffic(`C_a,C_b`)` function sorts the cores in the set C_a according to the communication traffic they have with the cores in the set C_b.

In the second phase, the mapping of cores c_i and c_j is inverted for each mapping $M\in\mathscr{M}$ and each pair (c_i,c_j), thus obtaining the new mapping M'. The algorithm proceeds with the next pair on the mapping M or M' according to whether M dominates M' or M' dominates M. If M and M' are Pareto mappings, the algorithm proceeds with the next pair on both mappings. A pseudo-code description of this phase is given in Figure 16, while Figure 17 describes the main program.

```
Mappings PBNMAP_2nd(Mappings M, Cores C)
{
   for (i ∈ {1,2,...,N²-1})
      for (j ∈ {i+1,i+2,...,N²}) {
         Mappings Mn = ∅;
         for (M ∈ M) {
            Mapping M' = Swap(M, i, j);
            if (M' ≻ M)
               Mn = Mn ⋃ {M'};
            else if (M ≻ M')
               Mn = Mn ⋃ {M};
            else
               Mn = Mn ⋃ {M,M'};
         }
         M = ExtraxtPareto(Mn);
      }

   return M;
}
```

Figure. 16: Second phase of the Pareto-based NMAP approach.

```
Mappings PBNMAP(Cores C)
{
   Mappings M;

   M = PBNMAP_1st(C);
   M = PBNMAP_2nd(M, C);

   return M;
}
```

Figure. 17: Pareto-based NMAP approach.

VI. Experiments

A. MPEG-4 Codec

In order to evaluate the various approaches in real traffic scenarios, an MPEG-4 simple profile @ level 2 codec was used as a case study [34]. A general block diagram of the encoder and decoder is shown in Figure 18.

For the hardware/software partitioning reference was made to the MoVa architecture described in [22]. It adopts a macroblock-based pipeline with 4 stages for the encoder and

Core	*Description*
MEC	Motion estimation coarse
MEF	Motion estimation fine
MC	Motion compensation
VLC	Variable length coding
VLD	Variable length decoding
REC	Reconstruction
SP	Stream producer
DB	Deblocking
DCTQ	Discrete cosine transform & quantization
IQIDCT	Inverse discrete cosine transform & inverse quantization
RISC	32 bit risc microprocessor
VIM	Video input module
VOM	Video output module
ISC	Input stream controller
MEME	Encoder memory
MEMD	Decoder memory

Table 1: Cores implementing the codec.

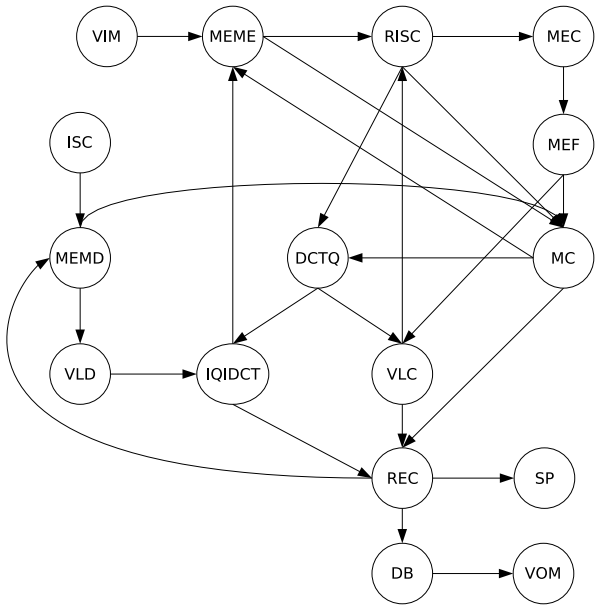


Figure. 19: Application characterization graph of the MPEG-4 codec.

3 for the decoder. More specifically, the encoding section performs coarse motion estimation in the first stage, fine motion estimation fine and motion compensation in the second stage, discrete cosine transform and quantization in the third stage, and finally reconstruction and production of the stream in the fourth stage. In the decoding section, the first stage involves variable length decoding of each data stream; in the second stage it performs sequential inverse cosine transformation, inverse quantization and motion compensation; the third and final stage is reconstruction.

To obtain the traffic traces the C application implementing the codec [24] was modified with the addition of a monitor code to record the volume of incoming and outgoing traffic in the various functional blocks into which the application is partitioned. Table 1 shows the 16 cores implementing the codec. They were characterized in terms of timing by using the clock cycle data in [22] for the execution of each operation (DCT, MC, etc.). For power characterization, we used the mean values given in the datasheets [27, 31]. For the interconnection system we used an approach similar to the one presented in [17]. To characterize the switches, a 5x5 switch was implemented in VHDL following the architecture described in [6]. It was synthesized with a Synopsys Design Compiler using the Virtual Silicon $0.13\mu m$, $1.2V$ technological library and analyzed using Synopsys Design Power using different random input data streams for the inputs of the switch. The amount of power consumed by a flit for a hop switch was estimated as being $0.181nJ$. We assumed the tile size to be $2mm \times 2mm$ and that the tiles were arranged in a regular fashion on the floorplan. The load wire capacitance was set to $0.50fF$ per micron, so considering an average of 25% switching activity the amount of power consumed by a flit for a hop interconnect is $0.384nJ$.

Figure 19 shows the application characterization graph of the MPEG-4 codec. Each vertex of the graph represents a core. An edge that connects a core i to a core j defines a communication flow from core i to core j. Each edge is characterized by a set of attributes such as the traffic volume ($T_{i,j}$) and the minimum bandwidth requirement for the communication ($B_{i,j}$). The latter one is used as an exploration constraint. More precisaly, a mapping is rejected if it does not satisfy at least one of such constraints. These constraints are set by performing a profiling of the application and annotating the traffic volume exchanged between the various application components. For example, to decode N frames at X fps we have $B_{i,j} = T_{i,j} \times X/N$.

The following values were used for the free parameters of the exploration algorithm. For *GAMAP* we chose a population of 50 mappings, a crossover probability of 0.7 and a mutation probability of 0.1. The R parameter of the crossover operator was set to 2. These values were chosen after numerous simulations and were the values that on average led to better solutions or shorter convergence times. The number of generations was set runtime by means of a stop criterion based on analysis of the convergence of the Pareto-front [4]. For *PBBB*, the parameter T_{pbbb} was set to 100. Figure 20 gives the power values and traffic clearing times for 10,000 random mappings. It also shows the Pareto fronts obtained by *GAMAP*, *PBNMAP*, and *PBBB*, and the solutions found by *BB* [17] and *NMAP* [28]. As can be seen, the solutions obtained by *GAMAP* dominate those obtained by the other approaches. The figure also shows the good trade-off between delay and power (respectively equal to a factor of 3 for delay and 2.5 for power).

Figure 21(a) gives the number of simulations (i.e. mappings evaluated by *GAMAP*) for varying numbers of generations. It gives the number of simulations actually performed and those virtually performed if no caching mechanism had been used. Figure 21(b) gives the normalized delay and en-

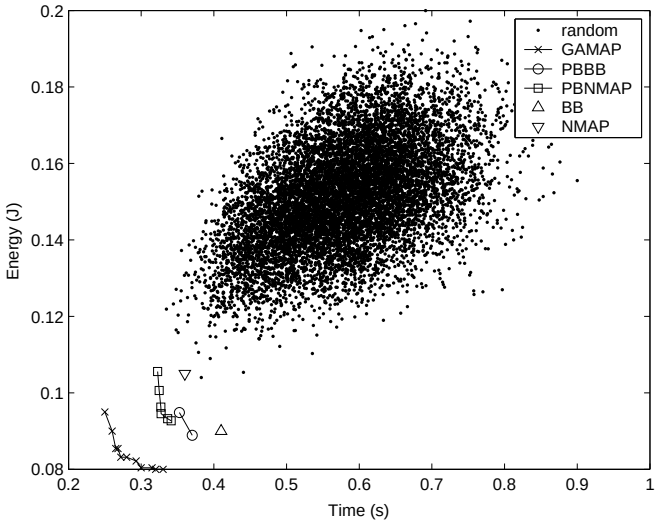


Figure. 20: Evaluation of 10,000 random mappings and Pareto fronts obtained by *GAMAP*, *PBNMAP*, and *PBBB* for a 4x4 NoC in which the MPEG-4 codec is mapped on.

ergy values for varying numbers of generations. As can be seen, in both cases no mappings that determine appreciable improvements in delay and energy consumption are found after the 20th generation. At the 20th generation *GAMAP* had only performed 840 simulations as compared with 2,670 by *PBNMAP* and 7,238 by *PBBB*, thus providing an exploration time speed-up of 3.2 and 8.6 respectively.

Encoder Decoder Shared

VOM	SP	MEME	VIM
DB	REC	IQ IDCT	VLD
MEC	RISC	MC	MEMD
MEF	VLC	DCTQ	ISC

Figure. 22: Pareto mapping for the l'MPEG-4 codec.

Finally, Figure 22 shows a point (the minimum energy mapping) in the Pareto set obtained by *GAMAP*. The cores specific to the encoding section are shown against a dark gray background, whereas those specific to the decoding are against a white background. The cores shared by the encoder and decoder are shown against a light gray background and have been mapped (in this case) in the centre of the NoC. In the decoding section, the cores VOM and DB are topologically separated from VLD, MEMD and ISC as there is no direct communication flow between these sets: they communicate by means of a ring represented by the core REC. In the encoding section there are also two separate parts which do not communicate directly but through the set of shared cores.

B. Cell Phone

Figure 23(a) is a block diagram of a mobile phone application in which it is possible not only to hold a normal conversation but also to listen to an MP3, surf the web, receive and send images, and listen to emails. The application example used is the *airport* scenario described in [30]. In this example the traffic flows are generated under certain synchronization constraints. For example, as can be seen from Figure 23(b), which shows a fragment of the communication timeline, it is not possible to read an email and perform MP3 streaming at the same time.

The application was partitioned into 13 cores [one for each block shown in Figure 23(a)] and mapped onto a 4×4 NoC. Cores for a concurrent synthesized application in which each core communicates at random with the others were mapped onto the remaining 3 tiles.

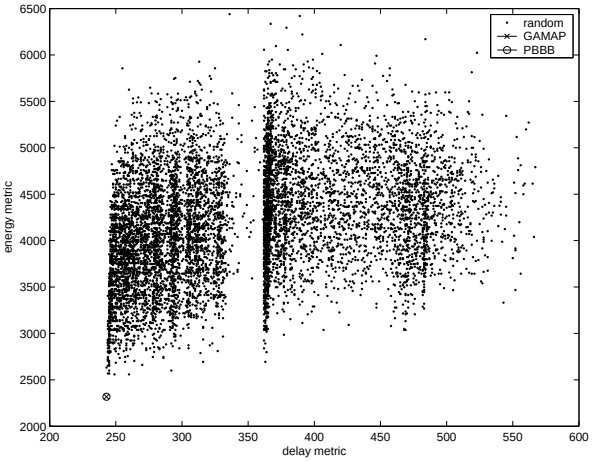


Figure. 24: Evaluation of 10,000 random mappings and Pareto fronts obtained by *GAMAP* and *PBBB* for a 4×4 NoC and cell phone application.

Figure 24 shows the solutions obtained by *GAMAP* and *PBBB* together with the evaluation of 10,000 randomly generated mappings. In this case it was not possible to complete the exploration using *PBNMAP* due to the great number of Pareto mappings obtained at each iteration during the first phase of the algorithm (Figure 15). The main reason for this behavior lies in the characteristics of the traffic considered. More specifically, in the first phase of the algorithm the mapping of a core that does not communicate with any of the other cores already mapped generates as many Pareto mappings as there are free tiles. In such situations the ExtractPareto($\mathscr{M}$) function returns the same set $\mathscr{M}$, the mappings of which will be extended in the following iteration by the MakeMappings($\mathscr{M}$, c) function to map the core c, generating a new set of mappings $|\mathscr{M}| \times f$ in size (where f indicates the number of free tiles in the incomplete mapping). Obviously, the more often this situation arises, the more quickly the number of mappings to be evaluated (and

thus the number of simulations to be performed) grows. In this example it happens quite frequently because the application was partitioned using a coarser granularity. The traffic flows, in fact, involve on average fewer cores than the previous examples, thus reducing the probability that a core being mapped will communicate with at least one of the cores already mapped.

Going back to Figure 24 we can observe a great range of dispersion between the points (2.3x for delay and 2.5x for energy consumption) which once again requires efficient techniques to explore the mapping space. In this example *GAMAP* and *PBBB* yield the same solution but the former requires only 1,227 simulations as compared with the 9,893 required by the latter.

VII. Conclusions

In this paper we have proposed a strategy for topological mapping of IPs/cores in a mesh-based NoC architecture. The approach uses heuristics based on multi-objective genetic algorithms to explore the mapping space and find the Pareto mappings that optimize performance and power consumption. At the same time, two of the most widely-known approaches to mapping in mesh-based NoC architectures have been extended in order to explore the mapping space in a multi-criteria mode. The approaches have been then evaluated and compared, in terms of both accuracy and efficiency, on a platform based on an un event-driven trace-based simulator which makes it possible to take account of important dynamic effects that have a great impact on mapping. The experiments carried out on real applications (an MPEG-4 encoder/decoder system and a cellular phone application) confirm the efficiency, accuracy and scalability of the proposed approach. Future developments will mainly address the definition of more efficient genetic operators to improve the precision and convergence speed of the algorithm. Evaluation will also be made of the possibility of optimizing mapping by acting on other architectural parameters such as routing strategies, switch buffer sizes, etc.

References

[1] C. J. Alpert, L. W. Hagen, and A. B. Kahng. A hybrid multilevel/genetic approach for circuit partitioning. In *Fifth ACM/SIGDA Physical Design Workshop*, pages 100–105, Apr. 1996.

[2] C. J. Alpert and A. B. Kahng. Recent developments in netlist partitioning: A survey. *VLSI Journal*, 19(1–2):1–81, 1995.

[3] ARM. AMBA specification. `http://www.arm.com/`, May 1999.

[4] G. Ascia, V. Catania, and M. Palesi. A GA based design space exploration framework for parameterized system-on-a-chip platforms. *IEEE Transactions on Evolutionary Computation*, 8(4):329–346, Aug. 2004.

[5] G. Ascia, V. Catania, and M. Palesi. Multi-objective mapping for mesh-based NoC architectures. In *Second IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis*, pages 182–187, Stockholm, Sweden, Sept. 8–10 2004.

[6] N. Banerjee, P. Vellanki, and K. S. Chatha. A power and performance model for network-on-chip architectures. In *Design, Automation and Test in Europe*, pages 1250–1255, Feb. 16–20 2004.

[7] M. S. Bright and T. Arslan. Synthesis of low-power DSP systems using a genetic algorithm. *IEEE Transactions on Evolutionary Computation*, 5(1):27–40, 2001.

[8] H. Chang, L. Cooke, M. Hunt, G. Martin, A. McNelly, and L. Todd. *Surviving the SOC Revolution A Guide to Platform-Based Design*. Kluwer Academic Publishers, 1999.

[9] J.-M. Chang and M. Pedram. Codex-dp: Co-design of communicating systems using dynamic programming. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 19(7):732–744, July 2000.

[10] C. A. C. Coello. A comprehensive survey of evolutionary-based multiobjective optimization techniques. *Knowledge and Information Systems. An International Journal*, 1(3):269–308, Aug. 1999.

[11] D. W. Corne, J. D. Knowles, and M. J. Oates. The pareto envelope-based selection algorithm for multiobjective optimization. In M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. J. Merelo, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature - PPSN VI 6th International Conference*, pages 839–848. Springer. Lecture Notes in Computer Science No. 1917, 2000.

[12] W. J. Dally and B. Towles. Route packets, not wires: On-chip interconnection networks. In *Design Automation Conference*, pages 684–689, Las Vegas, Nevada, USA, 2001.

[13] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, Apr. 2002.

[14] M. R. Garey and D. S. Johnson. *Computers and intractability: a guide to the theory of NP-completeness*. Freeman and Company, 1979.

[15] C. J. Glass and L. M. Ni. The turn model for adaptive routing. In *25 Years ISCA: Retrospectives and Reprints*, pages 441–450, 1998.

[16] A. Hemani, T. Meincke, S. Kumar, A. Postula, T. Olsson, P. Nilsson, J. Oberg, P. Ellervee, and D. Lundqvist. Lowering power consumption in clock by using globally asynchronous locally synchronous design style. In *ACM IEEE Design Automation Conference*, pages 873–878. ACM Press, 1999.

[17] J. Hu and R. Marculescu. Energy-aware mapping for tile-based NoC architectures under performance constraints. In *Asia & South Pacific Design Automation Conference*, Jan. 2003.

[18] J. Hu and R. Marculescu. DyAD - smart routing for networks-on-chip. In *ACM/IEEE Design Automation Conference*, pages 260–263, San Diego, CA, USA, June 7–11 2004.

[19] J. Hu and R. Marculescu. Energy- and performance-aware mapping for regular NoC architectures. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 24(4):551–562, Apr. 2005.

[20] IBM Corporation. The CoreConnect bus architecture. `http://www.ibm.com/`.

[21] K. Keutzer, S. Malik, R. Newton, J. M. Rabaey, and A. Sangiovanni-Vincentelli. System-level design: Orthogonalization of concerns and platform-based design. *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, 19(12):1523–1543, Dec. 2000.

[22] S.-M. Kim, J.-H. Park, S.-M. Park, B.-T. Koo, K.-S. Shin, K.-B. Suh, I.-K. Kim, N.-W. Eum, , and K.-S. Kim. Hardware-software implementation of MPEG-4 video codec. *ETRI Journal*, 25(6):489–502, Dec. 2003.

[23] V. Kommu and I. Pomenraz. GAFAP: Genetic algorithm for FPGA technology mapping. In *European Design Automation Conference*, pages 300–305, 1993.

[24] C. Lampert, M. Militzer, and P. Ross. XviD MPEG4 core library. `http://www.xvid.org/`.

[25] T. Lei and S. Kumar. A two-step genetic algorithm for mapping task graphs to a network on chip architecture. In *Euromicro Symposium on Digital Systems Design*, Sept. 1–6 2003.

[26] D. Martin, S. Ahmad, and K. Khalilian. Platform-based design: Report from the front. In *Ninth IEEE/DATC Electronic Design Processes Workshop*, Apr. 2002.

[27] Mentor Graphics. Inventra intellectual property cores. `http://www.mentor.com/inventra/cores/`.

[28] S. Murali and G. D. Micheli. Bandwidth-constrained mapping of cores onto NoC architectures. In *Design, Automation, and Test in Europe*, pages 896–901. IEEE Computer Society, Feb. 16–20 2004.

[29] Palmchip. SoC bus architecutre. `http://www.palmchip.com/`.

[30] J. M. Paul, D. E. Thomas, and A. Bobrek. Benchmark-based design strategies for single chip heterogeneous multiprocessors. In *Proceedings of the 2nd IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis*, pages 54–59, Stockholm, Sweden, Sept.8–10 2004. ACM Press.

[31] Philips Electronics. Philips' IP portfolio. `http://www.semiconductors.philips.com`.

[32] J. D. Schaffer. Multiple objective optimization with vector evaluated genetic algorithms. In *Genetic Algorithms and their Applications: Proceedings of the First International Conference on Genetic Algorithms*, pages 93–100. Lawrence Erlbaum, 1985.

[33] International thechnology roadmap for semiconductors. Semiconductor Industry Association, 2003.

[34] T. Sikora. The MPEG-4 video standard verification model. *IEEE Transactions on Circuits and Systems for Video Technology*, 7(1):19–31, Feb. 1997.

[35] Silicore. WISHBONE system-on-chip (SoC) interconnection architecture for portable IP cores. `http://www.silicore.net/`, Sept. 2002.

[36] Sonics. SiliconBackplane III MicroNetwork IP. `http://www.sonicsinc.com/`.

[37] D. Sylvester and K. Keutzer. Getting to the bottom of deep submicron. In *IEEE/ACM International Conference on Computer-aided design*, pages 203–211. ACM Press, 1998.

[38] D. Sylvester and K. Keutzer. A global wiring paradigm for deep submicron design. *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, 19(2):242–252, Feb. 2000.

[39] C. Valenzuela. A simple evolutionary algorithm for multi-objective optimization (SEAMO). In *Proceedings of the 2002 Congress on Evolutionary Computation CEC2002*, pages 717–722, 2002.

[40] C. L. Valenzuela and P. Y. Wang. VLSI placement and area optimization using a genetic algorithm to breed normalized postfix expressions. *IEEE Transactions on Evolutionary Computation*, 6(4):390–401, 2002.

[41] D. A. V. Veldhuizen and G. B. Lamont. Multiobjective evolutionary algorithms: Analyzing the state-of-the-art. *Evolutionary Computation*, 8(2):125–147, 2000.

[42] VSI Alliance. On-chip bus attributes specification version 1. `http://www.vsi.org/`, Sept. 2001.

[43] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the performance of the strength pareto evolutionary algorithm. In *EUROGEN 2001. Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems*, pages 95–100, Athens, Greece, Sept. 2001.

[44] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach. *IEEE Transactions on Evolutionary Computation*, 4(3):257–271, Nov. 1999.

Author Biographies

Giuseppe Ascia received the Laurea degree in electronic engineering and the Ph.D. degree in computer science from the Universit di Catania, Italy, in 1994 and 1998, respectively. In 1994, he joined the Institute of Computer Science and Telecommunications at the Universit di Catania. Currently, he is an Associate Professor at the Universit di Catania. His research interests are soft computing, VLSI design, hardware architectures, and low-power design.

Vincenzo Catania received the Laurea degree in electrical engineering from the Universit di Catania, Italy, in 1982. Until 1984, he was responsible for testing microprocessor system at STMicroelectronics, catania, Italy. Since 1985 he has cooperated in research on computer network with the Istituto di Informatica e Telecomunicazioni at the Universit di Catania, where he is a Full Professor of computer science. His research interests include performance and reliability assessment in parallel and distribuited system, VLSI design, low-power design, and fuzzy logic.

Maurizio Palesi received the Dr.Eng. degree and the Ph.D. degree in computer engineering from Universit di Catania, Italy, in 1999 and 2003 respectively. Since December 2003, he has held a research contract as Assistant Professor at the Dipartimento di Ingegneria Informatica e delle Telecomunicazioni, Facolt di Ingegneria, Universit di Catania. His research focuses on Platform based system design, design space exploration, low-power techniques for embedded systems, and Network-on-Chip architectures.

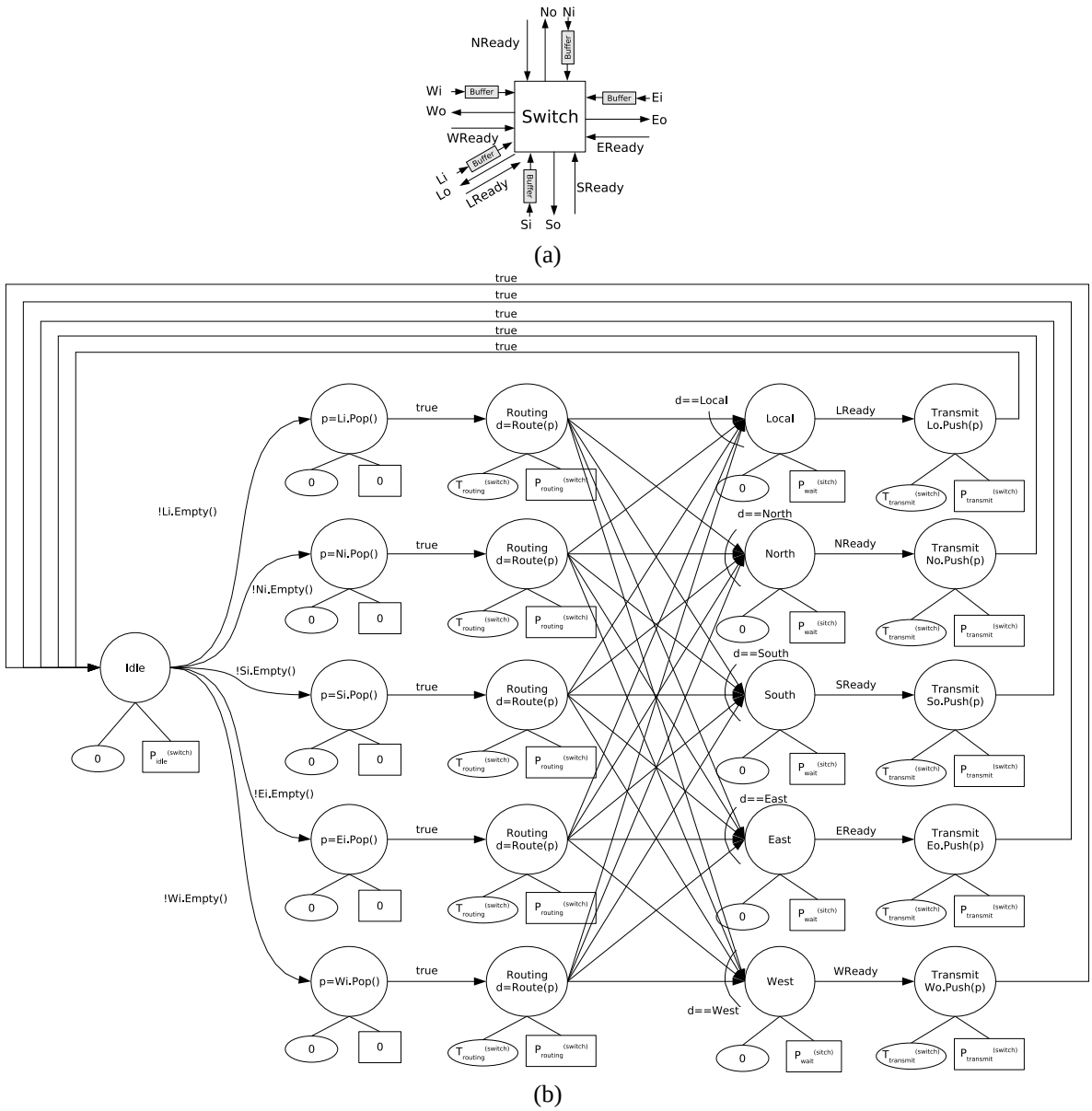
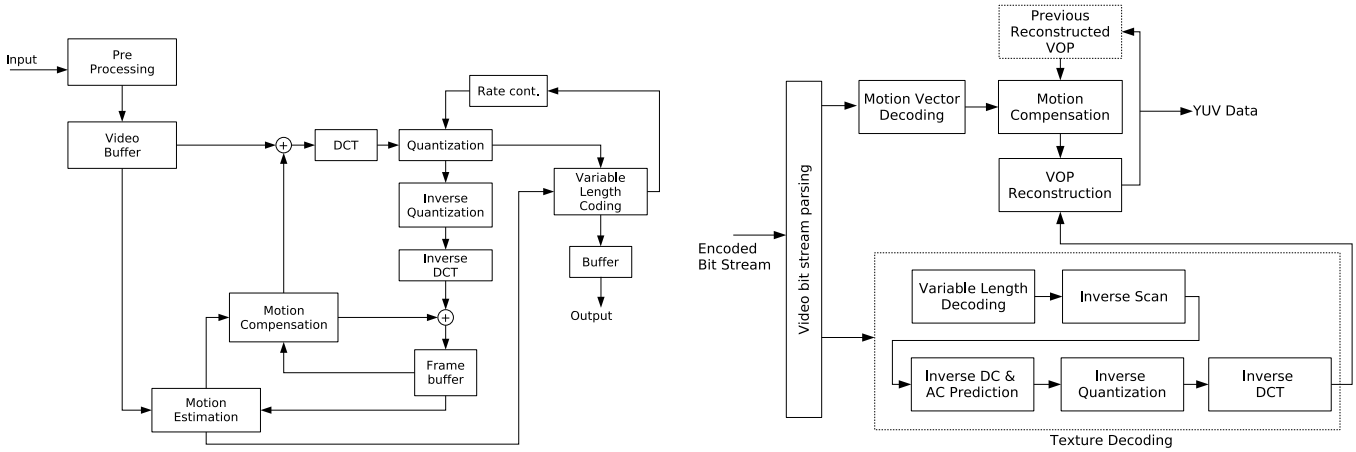
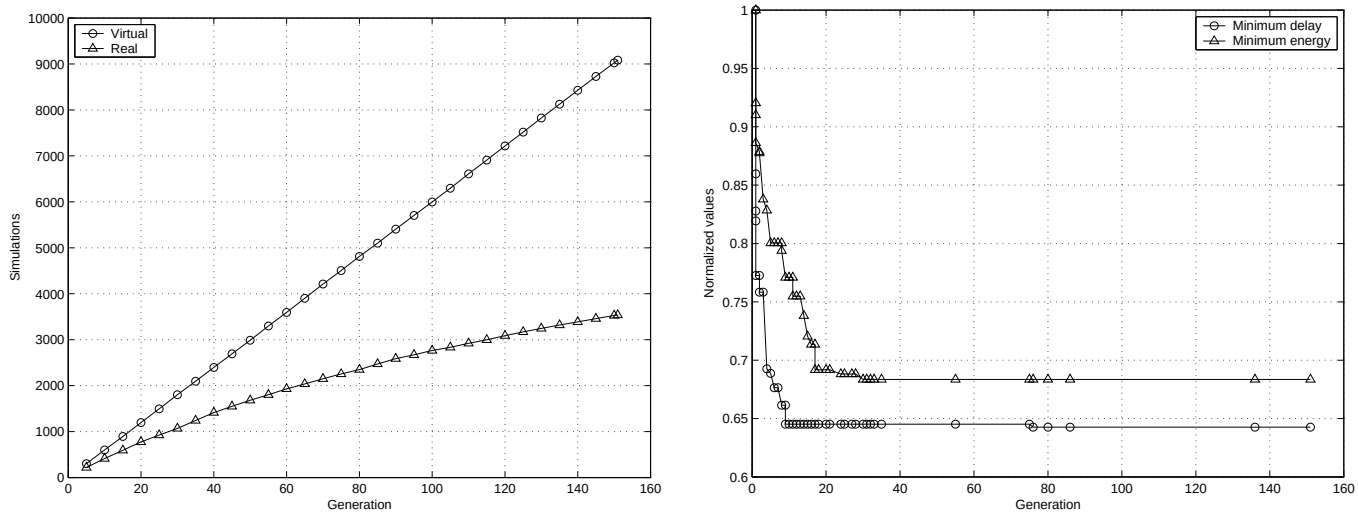


Figure. 4: Switch interface (a), Behavioural annotated graph of a switch (b).



(a) (b)

Figure. 18: Block diagram of the MPEG-4 codec. (a) Encoder. (b) Decoder.



(a) (b)

Figure. 21: Number of (virtual and real) mappings evaluated by *GAMAP* in varying numbers of generations (a). Normalized minimum delay and power consumption values obtained by the *GAMAP* in varying numbers of generations (b).

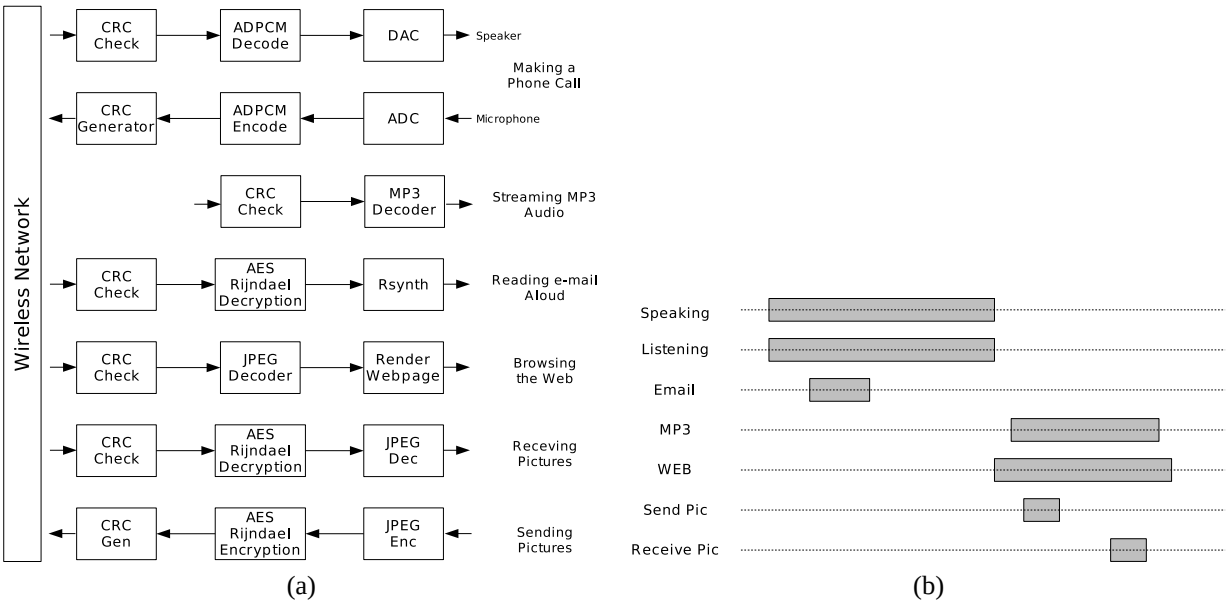


Figure. 23: Example application of a cell phone (source [30]) (a). A portion of the communication timeline (b).