Multi-objective Mapping for Mesh-based NoC Architectures

Giuseppe Ascia
Dipartimento di Ingegneria Informatica e delle Telecomunicazioni
University of Catania, Italy
gascia@diit.unict.it

Vincenzo Catania
Dipartimento di Ingegneria Informatica e delle Telecomunicazioni
University of Catania, Italy
vcatania@diit.unict.it

Maurizio Palesi
Dipartimento di Ingegneria Informatica e delle Telecomunicazioni
University of Catania, Italy
mpalesi@diit.unict.it

ABSTRACT

In this paper we present an approach to multi-objective exploration of the mapping space of a mesh-based network-on-chip architecture. Based on evolutionary computing techniques, the approach is an efficient and accurate way to obtain the Pareto mappings that optimize performance and power consumption. Integration of the approach in an exploration framework with a kernel based on an event-driven trace-based simulator makes it possible to take account of important dynamic effects that have a great impact on mapping. Validation on both synthesized traffic and real applications (an MPEG-2 encoder/decoder system) confirms the efficiency, accuracy and scalability of the approach.

Categories and Subject Descriptors

B.4.3 [**Input/Output and Data Communications**]: Interconnections (Subsystems); I.6.7 [**Simulation and Modeling**]: Simulation Support Systems—*Environments*; G.1.6 [**Numerical Analysis**]: Optimization

General Terms

Performance, Design

Keywords

Network-on-chip, mapping, multi-objective optimization, genetic algorithms, simulation.

1. 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-verified 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. Projections show that synchronous regions will occupy an increasingly lower fraction of a chip [13] giving rise to locally synchronous, globally asynchronous solutions [7]. Applications will be modeled as a set of communicating tasks with different characteristics and origins, which will make implementations extremely heterogeneous. 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 [3]. 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 standardizing on-chip communications. In this paper we will focus on mesh-based NoC architectures, in which resources communicate with each other via a mesh of switches that route and buffer messages. A resource is generally any core: a processor, a memory, an FPGA, a specific hardware block or any other IP compatible with the NoC interface specifications.

One of the most onerous tasks in this context is the topological mapping of the resources on the mesh in such a way as to optimize certain performance indexes (e.g. power, performance). It is therefore of strategic importance to define methods to search for a mapping that will optimize the desired performance indexes.

The problem of mapping in mesh-based NoC architectures has been addrees in three previos papers. Hu and Marculescu [8] 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. Murali and De Micheli [11] 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 [9] present an approach that uses genetic algorithms to map an application, described as a parameterized task graph, on a mesh-based NoC architecture 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, such as the variation in delay due to the switch input buffers which, as we will see, have a great impact on choice of the mapping. In agreement with [12], we believe that analytical methods make too many assumptions about the network and traffic to get accurate values for real systems. The second problem relates to the optimization method used. It refers in all cases to a single performance index (power in [8], performance in [11, 9]). As we will see in the section devoted to experiments, optimization of one performance index may lead to unacceptable values for

CODES+ISSS'04, September 8–10, 2004, Stockholm, Sweden.

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.

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. The mappings visited during the exploration process will be evaluated using a simulation-based approach and the optimization objectives will be performance and power consumption. The simulation-based approach makes it possible to evaluate the impact of the main architectural and application parameters on performance and power.

The rest of the paper is organized as follows. Section 2 presents the simulation and evaluation framework used. The impact of the architectural and application parameters on the performance indexes considered is assessed in Section 3. Section 4 presents the algorithm for exploration of the mapping space and evaluates it in terms of accuracy and efficiency in different traffic scenarios. Finally, Section 5 summarizes our contribution and outlines some directions for future work.

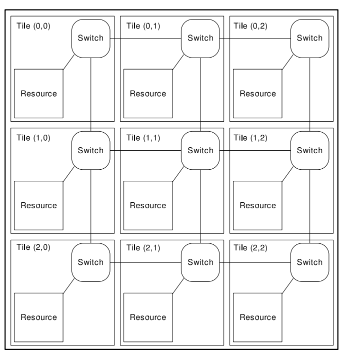

Figure 1: Structure of a 3x3 mesh-based NoC architecture.

2. EVALUATION OF A MAPPING

Figure 1 shows the NoC topology we will refer to. It is a two-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. In this section we will describe the simulation and power and delay estimation model used to evaluate a mapping.

Each switch in the NoC is connected to the four adjacent switches except for those at the network boundaries. 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 packets to be routed are queued. The routing algorithm features *static XY* routing in which a packet 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). The point-to-point connections between two switches and between a switch and a resource are taken to be 256-bit (which corresponds to the size of a packet). As a transmission scheme we use wormhole routing.

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* (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 energy consumption is calculated as the product between *power* and the time spent in the state. 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 packet in the input queue it passes to a fictitious state (featuring $latency = 0$ and $power = 0$) in which the type of packet is evaluated. If it is the first in a data flow directed towards the core involved (head packet, H) or an intermediate packet (body packet, 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 packet basis but on a set of data that cannot be contained in a single packet. If, on the other hand, the packet is the last in a communication flow, the core switches to the *Process* state in which the packet 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 packet at the head of the queue and then, when the latency time ends, to switch unconditionally back to the *Idle* state.

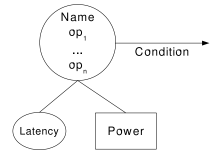

Figure 2: Behavioural annotated graph (BAG).

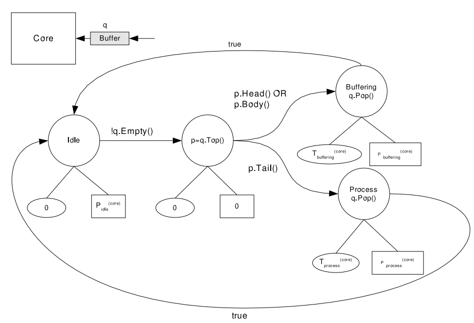

Figure 3: Behavioural annotated graph of a generic core.

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 packet. 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 packet in at least one of the 5 input queues the switch passes to a fictitious state (with $latency = 0$ and $power = 0$) in

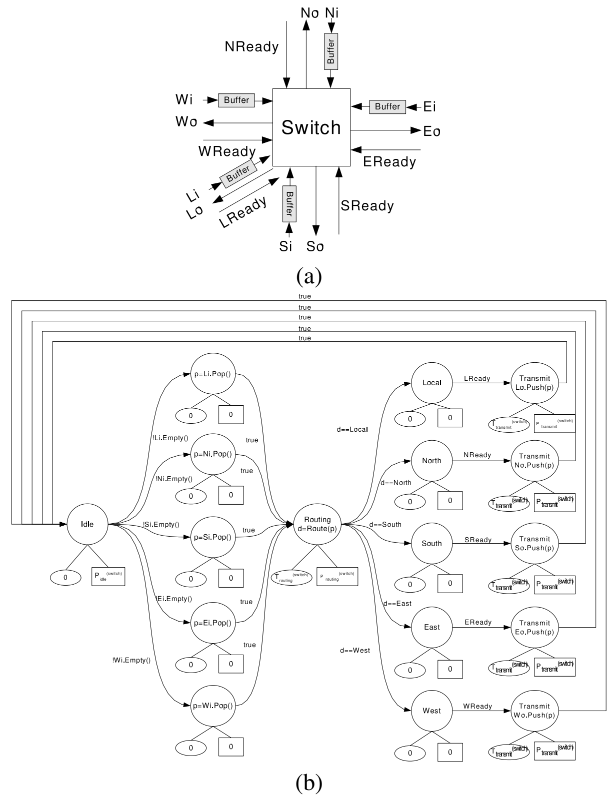

Figure 4: Switch interface (a), Behavioural annotated graph of a switch (b).

which the packet is read and immediately afterwards to the *Routing* state. In this state (which features an average power consumption of $P^{(switch)}_{routing}$ and a latency of $T^{(switch)}_{routing}$) the output port on which to route the packet is determined and the relative fictitious state (*LOCAL, NORTH, SOUTH, EAST, WEST*) is entered. Only when the packet is ready to be transmitted (*ready=true*) does the switch pass to the *Transmit* state in which the packet at the head of the input queue is extracted and then, on expiry of the latency time $T^{(switch)}_{transmit}$, unconditionally returns to the *Idle* state, with an average power consumption of $P^{(switch)}_{transmit}$, 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.

3. 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 our 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 shows the time required to handle traffic versus the size

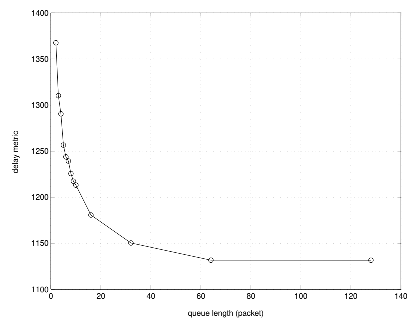

Figure 5: Traffic draining time vs. switch input queue size.

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 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.

It can, however, be observed that in some cases an increase in the

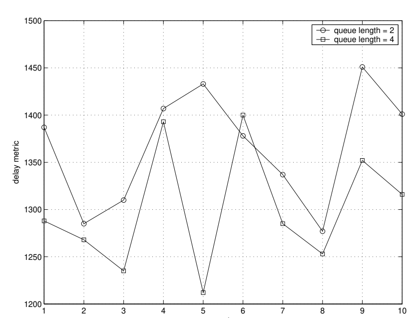

Figure 6: Traffic draining time for 10 different mappings and two different networks (with switch input queues of 2 and 4 packets).

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 char-

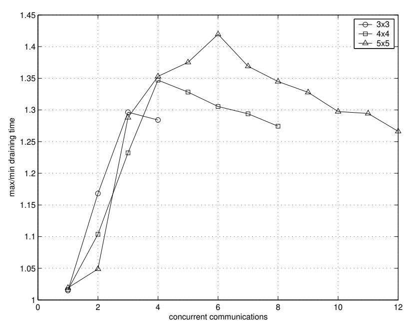

Figure 7: Relation between maximum and minimum traffic draining time for 1,000 random mappings with varying numbers of concurrent communications and different network sizes.

acteristics 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 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.

4. MULTI-OBJECTIVE EXPLORATION OF THE MAPPING SPACE

In this section we will describe our proposal for multi-objective exploration of the mapping space. To the best of our knowledge, our work is the first to address the mapping problem in an multi-objective fashion. The approach uses evolutionary computing techniques, specifically genetic algorithms (GAs), to obtain an approximation of the Pareto-optimal performance/energy front.

4.1 Exploration Framework

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

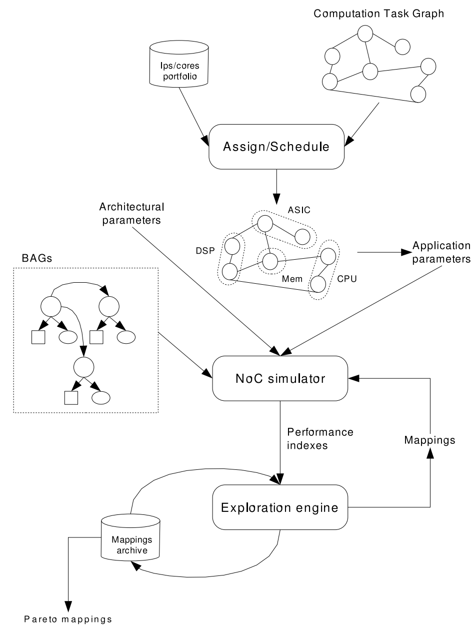

Figure 8: Framework for simulation and exploration of the mapping space.

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.

- *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.

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 exploration algorithms.

4.2 GA-based Multi-objective Exploration of the Mapping Space

When the exploration space is too vast to be explored exhaustively, a solution is to use evolutionary computing techniques. Genetic Algorithms (GAs) have found application in various VLSI design environments [10]: in problems relating to layout such as partitioning, placement and routing; in design problems including power estimation, low-power synthesis, technology mapping and netlist partitioning and in reliable chip testing through efficient test vector generation. All these problems are untreatable, in the sense that no polynomial time algorithm can guarantee an optimal solution.

In this paper we propose to use GAs for multi-objective mapping space exploration in a mesh-based NoC architecture in order to obtain an accurate approximation of the Pareto-optimal front of the mappings that optimize performance and power consumption. The approach is general because the solution to any problem using GAs only requires definition of a representation of the configuration, genetic operators and objective functions to be optimized. It is also an efficient approach because, as we shall see below, it only needs to visit a very limited number of configurations to provide an accurate approximation of the Pareto-optimal set. More specifically, we chose SPEA2 [14], which is very effective in sampling from along the entire Pareto-optimal front and distributing the solutions generated over the trade-off surface.

To apply a GA to the problem being examined, it is necessary to define the chromosome and genetic operators. The chromosome is a representation of the solution to the problem, which in this case is described by the mapping. Each tile in theNoC has an associated gene which encodes the identifier of the core mapped in the tile. In an $n \times m$ NoC, 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 $i\%n$ (where the symbol % indicates the modulus operator). We use single-point-crossover and mutation to generate the next new population. More specifically, the function performed by the genetic operators is to remap hot spot cores in a random fashion (a hot spot core being one whose switch has a greater average buffer occupation). The definition of suitable and more effective genetic operators has a great impact on the results of the optimization. This is not, however the aim of this paper and remains a topic for future research.

4.3 Experiments

We will start by evaluating the performance of the exploration strategy on a *synthesized* case, i.e. one in which the traffic is generated statistically without particular reference to a specific application. We considered a 4x4 network and 8 traces of concurrent communications. The statistical distribution of the destination addresses was randomly chosen for each trace, and the size of the data flow exchanged between two nodes had a Gaussian distribution with an average of 128 bytes and a variance of 64 bytes. The attributes of the BAGs for the cores and switches were chosen at random between 0 and 1 and four input queues were assigned to each switch. As it is computationally unfeasible to explore each possible mapping exhaustively, the solutions obtained using the proposed approach were compared with those obtained by randomly sampling the mapping space in 200,000 different points.

Figure 9 shows the 200,000 random mappings evaluated and the Pareto fronts obtained by the proposed approach after 10, 20, 50 and 100 generations. As can be seen, after only 10 generations (using a population size and archive size of 50 and 10 elements respectively), the solutions obtained are not dominated by any of the 200,000 random mappings. The solutions obtained after 10 generations are better than those obtained by evaluating 200,000 random mappings. 10 generations required evaluation of only 680 mappings, i.e. less than 0.7%, with a consequent speedup of 147.

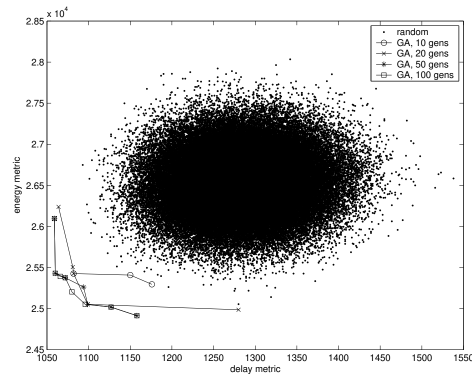

Figure 9: Evaluation of 200,000 random mappings and Pareto fronts obtained using the proposed approach.

To evaluate the proposed approach on a real case study, it was applied to a system comprising an MPEG-2 encoder and decoder [1]. The encoder was subdivided into 9 separate tasks and the decoder into 4 tasks, communicating with each other using a shared memory. They were assigned and scheduled on 12 IPs [4, 5, 6] comprising DSPs, generic processors, embedded DRAMs and customized ASICs. Figure 10 shows the partitioning of the application and assignment of the IPs.

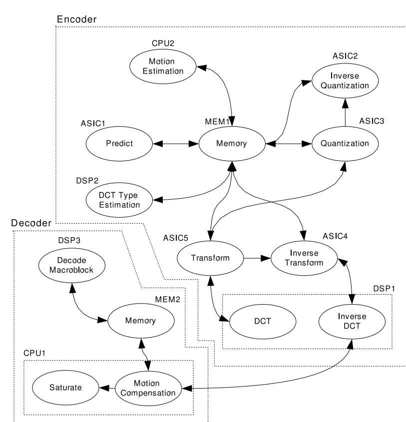

Figure 10: Partitioning of the MPEG-2 (encoder and decoder) application.

The traffic traces were obtained directly by executing the encoder-/decoder application on the first 5 frames of two different movie clips. The application was partitioned by identifying the parts that were to be implemented by the IPs chosen and then adding monitoring code in order to capture the inter-communication traffic between the IPs. The data for characterization of the IPs were taken from their respective datasheets. To characterize the switches, a 5x5 switch was implemented in VHDL following the architecture described in [2]. It was synthesized with Synopsys Design Compiler using the Virtual Silicon $0.13\mu m$, $1.2V$ technological library

High performance

	0	1	2
0	ASIC2	ASIC5	ASIC1
1	ASIC3	ASIC4	MEM1
2	**DSP3**	DSP2	CPU2
3	**MEM2**	**CPU1**	DSP1

Execution time: 40.6 ms
Energy: 18.9 mJ
(a)

Low energy

	0	1	2
0	ASIC2	ASIC5	ASIC1
1	ASIC3	ASIC4	MEM1
2	**CPU1**	DSP1	CPU2
3	**MEM2**	**DSP3**	DSP2

Execution time: 41.4 ms
Energy: 16.2 mJ
(b)

Figure 12: High performace mapping (a), and low energy mapping (b).

and analyzed using Synopsys Design Power.

Figure 11 shows the power and execution time values for 100,000

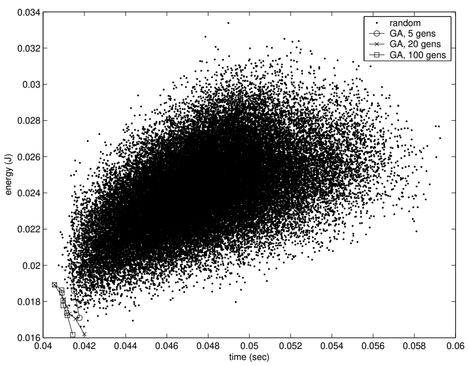

Figure 11: Evaluation of 100,000 random mappings and Pareto fronts obtained using the proposed approach on a 4x3 NoC in which an MPEG-2 encoder and decoder are mapped.

random mappings. It also gives the Pareto front of the mappings obtained by the proposed approach after 5, 20 and 100 generations. As can be seen, after only 20 generations (which correspond to just over 1,000 simulations), the solutions obtained by the proposed approach are not dominated by any of the 100,000 random mappings and are very close to those obtained after 100 generations.

Figure 12 shows the two mappings relating to the end points of the Pareto front which respectively determine maximum performance and minim power consumption. This example clearly shows the usefulness of multi-objective exploration Mono-objective exploration aiming at optimizing performance alone, for example, would have produced the mapping shown in Figure 12(a), neglecting the mapping in Figure 12(b) which, although it performs 2% less well, is 17% more efficient from the point of view of power consumption. In Figure 12 the cores for the decoder subsystem are shown in bold type. It is interesting to note that the two subsystems are mapped in two disjoint partitions of the mesh. The genetic algorithm generally partitions the mesh into highly interactive clusters of cores and then optimizes the positioning of the cores in each cluster. At the same time it also optimizes inter-cluster communications, as can be seen by observing the two mappings shown in Figure 12. In both cases DSP1, which implements calculation of the DCT and IDCT, is mapped on the boundary between the two partitions as the core is used by both subsystems.

5. 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. The experiments carried out on both synthesized traffic and real applications (an MPEG-2 encoder/decoder system) confirm the efficiency, accuracy and scalability of the 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.

6. REFERENCES

[1] I. D. 13818-2. MPEG-2 video. ISO standard, 1994.

[2] 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.

[3] 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.

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

[5] Altera. Altera intellectual property: IP megastore. `http://www.altera.com/products/ip/`.

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

[7] 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.

[8] 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.

[9] 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.

[10] P. Mazumder and E. M. Rudnick. *Genetic Algorithms for VLSI Design, Layout & Test Automation*. Prentice Hall, Inc., 1999.

[11] 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.

[12] S. G. Pestana, E. Rijpkema, A. Radulescu, K. Goossens, and O. P. Gangwal. Cost-performance trade-offs in networks on chip: A simulation-based approach. In *Design, Automation, and Test in Europe*, pages 896–901. IEEE Computer Society, Feb. 16–20 2004.

[13] 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.

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