Evaluation of Algorithms for Low Energy Mapping onto NoCs

César A. M. Marcon, Edson I. Moreno, Ney L. V. Calazans and Fernando G. Moraes
Faculty of Informatics - Pontifícia Universidade Católica do Rio Grande do Sul
Porto Alegre, RS – Brazil
e-mail: {marcon, emoreno, calazans, moraes}@inf.pucrs.br

***Abstract*—Systems on Chip (SoCs) congregate multiple modules and advanced interconnection schemes, such as networks on chip (NoCs). One relevant problem in SoC design is module mapping onto a NoC targeting low energy. To date, few works are available on design and evaluation of mapping algorithms. The main goal of this work is to propose some algorithms and evaluate its results and performance with regard to low energy NoC mappings. These include exhaustive and stochastic search methods and heuristic approaches, and some combinations. The use of combined approaches compared to pure stochastic algorithms provides average reductions above 98% in execution time, while keeping energy saving within at most 5% of the best results. In addition, one heuristic provided average reductions in execution time above 90% when compared to pure stochastic algorithms, and obtained better energy saving than combined approaches.**

I. Introduction

A NoC is an intra-chip communication infrastructure, composed by routers interconnected by communication channels. In direct NoC topologies, each router connects to a module and both are placed inside a limited region called *tile*. NoCs are seen as promising communication resources for implementing future SoCs. Consider an application composed by **n** modules, such as DSPs, other processors and memories, implemented as a SoC. Assume this SoC employs a NoC as communication infrastructure. The *module mapping problem* consists in finding a mapping of each module to a tile to optimize a cost function (e.g. energy consumption, performance). Generally, this mapping problem allows ***n!*** possible solutions. Given future SoCs with hundreds of tiles [1], exhaustive search of the solution space is unfeasible. Thus, the optimal implementation of such SoCs requires more efficient mapping approaches.

The main goal of this work is to evaluate algorithms for the module-mapping problem, having energy consumption as cost function, given that communication can highly affect it [2]. The evaluated algorithms include from exhaustive and stochastic search to heuristic approaches, plus some combinations of these.

The remaining of this paper is organized as follows. Section II presents the problem definition. Section III discusses related work. Section IV shows how to model application and architecture, while sketching the algorithms. Section V shows results and Section VI presents some conclusions and future work.

II. Problem Formulation

Definition 1: A *Communication Weight Graph* (CWG) is a directed graph $CWG = <M, C>$, where $M = \{m_1, m_2, \ldots, m_n\}$ represents the set of modules, corresponding to the set of CWG vertices, and $C = \{(m_i, m_j, W_{ij}) \mid (m_i, m_j) \in M\}$ denotes the set of communications between modules, corresponding to the set of CWG edges. The edge *weight* W_{ij} in (m_i, m_j, W_{ij}) represents the total data amount in bits, sent from m_i to m_j.

Definition 2: A *Communication Resource Graph* (CRG) is a directed graph $CRG = <\Gamma, L>$, where $\Gamma = \{\tau_1, \tau_2, \ldots, \tau_p\}$ denotes the set of tiles, corresponding to the set of CRG vertices. $L = \{(\tau_i, \tau_j) \mid \tau_i, \tau_j \in \Gamma\}$ designates the set of routing paths from tile τ_i to tile τ_j, corresponding to the set of CRG edges. The number of NoC tiles is denoted by $\boldsymbol{p}$.

CWG represents the *communication weight*, i.e. the number of phits[1] transmitted between application modules. CRG represents the *communication resources* of the target architecture.

III. Related Work

Table I reviews some of relevant works on the module mapping problem. Column *Basic element* shows if application behavior is modeled through modules or tasks interrelation. Column *Model* exposes the variety of application models to represent the *quantity*[2] and/or the *order*[3] of computation and/or communication. Column *Algorithm* also exposes a great range of possibilities. Column *Objective function* shows that energy and latency are the most evaluated requirements. Column *Benchmark* exposes that synthetic and embedded applications have been preferred for evaluation. Column *Topology* summarizes that NoC 2D mesh is also preferred. Finally, column *Maximum size (tiles)* shows that the majority of target architectures have less than 100 tiles.

Following the reviewed works trend, this paper analyzes mapping algorithms with respect to the energy consumption of 2D mesh NoCs with XY routing algorithm. Applications were described with communication quantity, using a communication weight model (CWM) [3].

IV. Mapping Algorithms

This work uses the CAFES framework [18], since it supports application modeling (e.g. CWM), mapping algorithms and syn-

[1] *Phit* is the physical unit information sent in a channel.
[2] *Quantity* information allows modeling e.g. the number of bits transmitted in a single communication.
[3] *Order* can be partial or total. When order is partial, it is called *dependence*. Dependence allows knowing e.g. all messages that a given message depends on. A *total ordering* enables to know e.g. the exact instant of time a given task will be executed.

thetic application generation. Based on the target architecture and the description of an application partitioned in modules, CAFES also allows obtaining the CRG and CWG, which are inputs to mapping algorithms.

TABLE I. ACCOUNT OF RELATED WORK IN MAPPING ALGORITHMS AND APPLICATION MODELS FOR NOC COMMUNICATION ARCHITECTURES

[a]Work (Year)	Application		Mapping		Benchmark	NoC	
	Basic element	Model	Algorithm	Objective function		Topology	Max size (tiles)
[3] / (2005)	module	communication dependence and quantity	exhaustive and SA	energy	embedded and synthetic	mesh	100
[4] / (2005)	module	communication quantity	branch-and-bound	energy and bandwidth	video, audio and synthetic	mesh	169
[5] / (2004)	module	communication quantity	branch-and-bound	bandwidth		mesh and torus	65
[6] / (2005)	module	communication dependence and quantity, and computation quantity	exhaustive and SA	energy and latency	embedded and synthetic	mesh	120
[7] / (2005)	module	communication quantity and total ordering	tabu search	energy and latency	embedded and synthetic	mesh, folded-torus and fat tree	16
[8] / (2004)	module	communication quantity	genetic	energy and latency	embedded and synthetic	mesh	25
[9] / (2004)	module	communication quantity	genetic	thermal balance	codec	tile-based NoCs	100
[10] / (2006)	task	computation dependence and quantity	exhaustive	area and latency	multiprocessors and synthetic	mesh	9
[11] / (2003)	task	computation dependence and quantity	genetic	latency	synthetic	mesh	15
[12] / (2004)	module	communication quantity	integer linear programming	energy	video and synthetic	mesh	16
[13] / (2003)	task	computation dependence and quantity	genetic	energy	embedded and synthetic	-	5
[14] / (2005)	module	communication quantity	greedy and stochastic	energy and latency	video	mesh, torus, butterfly, clos and hypercube	16
[15] / (2005)	module	communication quantity	branch-and-bound	energy	synthetic	mesh	40
[16] / (2004)	task	computation dependence and quantity	greedy	energy and deadline	synthetic	mesh	16
[17] / (2005)	module	communication quantity	greedy and tabu search	latency, energy, QoS area and critical constraints	video	mesh, torus, butterfly, clos and hypercube	16
[18] / (2005)	module	communication quantity	greedy	energy, bandwidth and latency	embedded and multimedia	mesh	16

Mapping algorithms are divided in two parts: (*i*) an internal part, which is target architecture and application model dependent; and (*ii*) an external part that generates mappings and calls the internal part to compute the mapping cost. Four basic external algorithms are evaluated for solving the module-mapping problem: exhaustive search (ES), simulated annealing (SA), tabu search (TS) and two greedy heuristics proposed here: Largest Communication First (LCF) and Incremental (GI). From these, the Authors derived three mixed approaches: SA and TS with seed mapping generated by LCF (HSO and HT, respectively), and a version of SA with a single iteration loop (HSM), using LCF seed mapping. Since ES has a trivial formulation, it is not discussed here.

A. Simulated Annealing (SA) Algorithm

SA is a stochastic approach implementing the external mapping algorithm with two nested loops. The external loop implements a global search technique, and the internal loop implements a local refinement. The internal loop optimizes mappings provided by the external one, successively swapping two random modules and storing the best mapping thus obtained. Best mapping is defined as the mapping with lowest energy consumption when compared to previously computed mappings. To avoid local minima, searches that start from swaps that do not save energy can be accepted depending on a *temperature* parameter. Larger temperatures imply greater probability to accept worse mappings. After each execution of the internal loop, temperature is decremented. The final best mapping from the internal loop is compared to the global best mapping and the best energy saving is stored as a new best mapping. The external loop randomly swaps several modules to search for widely different mappings.

B. Tabu Search (TS)

Like SA, TS is a stochastic approach implementing the external mapping algorithm with two nested loops. The internal loop randomly searches pairs of modules, looking for a swap that results in less energy consumption. A *swap list* with all distinct pairs of swappable modules optimizes the search. E.g., if the list contains a pair (R_x, R_y), it will not contain the pair (R_y, R_x). The search process accesses the swap list randomly. If a swap saves more energy than a previous one, the *swap list* index of this pair is stored. When the internal loop finishes, if an energy saving pair was found, it is inserted into a *tabu list*, removed from the *swap list*, and the current mapping is modified with this swap. If not, no action is taken and a new internal loop is performed.

C. Largest Communication First (LCF)

LCF is a greedy heuristic proposed here, where the *most communicating* modules are mapped first. The amount of communication is computed for each module using a balance between communication weight (associated to CWG edges) and the number of tiles with which each module communicates. Here, modules are placed from the center to the borders of the communication infrastructure.

Six steps constitute LCF: (*i*) creation of a list containing the NoC router types that associates module communication needs with router communication capacity; (*ii*) conversion of CWG to an undirected graph, combining the weight of opposed parallel edges. This increases performance of the algorithm, while keeping its accuracy; (*iii*) associating modules to routers, considering communication weight and amount of modules with which the module communicates; (*iv*) creation of a *communication list* based on the undirected graph. The list is ordered decreasingly, according to the amount of communication between pairs of modules; (*v*) using the *communication list* to assign modules to router types, creating an *assignment list*, considering module communication requirement and router availability; (*vi*) mapping creation, considering the *assignment list* and the place of each router type. This guarantees that modules placed at the center of the NoC are those with largest communication demands.

D. Greedy Incremental (GI) Algorithm

GI is also a 2-nested loops algorithm. Given a predefined randomly generated initial mapping, the external loop fixes a place to be the *pivot* of evaluation. The internal loop performs swaps using as decision criterion the largest obtained energy saving.

Five steps compose the GI algorithm: (*i*) CWG conversion, equal to step (*ii*) of LCF. (*ii*) Initial mapping definition. (*iii*) Choosing a NoC position to be the pivot for internal loop evaluation. The pivot place changes incrementally each time the external loop is executed. The first pivot position is the top-left corner of the NoC and the last is the bottom-right corner of the NoC. (*iv*) In-

ternal loop execution defining which swap leads to the best energy saving. It operates by incrementally computing savings achieved by swapping the chosen position module with each module in one of the remaining positions. These places include only those not yet chosen by the external loop as pivots. When the algorithm attains the end of the internal loop, it has achieved the most energy saving alternative among all possibilities evaluated up to here. (*v*) If there is a pair of NoC positions leading to a better mapping, this step swaps them. Otherwise nothing occurs. After the action, the algorithm discards the pivot from consideration. Next, execution resumes to step (*iii*), to define another pivot. The algorithm stops when there are no more pivots available. The greedy nature of the algorithm is defined by the discard policy described when discussing step (*iv*).

E. Mixed Approaches (HSO, HT, HSM)

The initial mapping used in stochastic algorithms influences the search for optimal mappings. Since LCF is the fastest presented approach and obtains figures better than random mappings (See Section V), it is a potentially good generator of seed mappings. Three approaches help enhancing stochastic algorithms with LCF: (*i*) First, HSO mixes LCF and SA. While SA external loop randomly swaps several modules to produce widely different mappings, HSO mappings are produced only once by LCF. Thus, HSO performs refinement by swapping modules in the internal loop, looking for local minima. The last mapping resulting from the internal loop execution is input for the next internal loop. (*ii*) Second, HT mixes LCF and TS. Similar to HSO, the random seed mapping generation inside TS is replaced by an LCF-generated mapping. The remaining of TS is unaltered. (*iii*) Third, HSM mixes LCF and SA. It differs from HSO since the external loop is performed just once, which may lead to worse mappings, but drastically reduces execution time.

V. EXPERIMENTATION

Mapping algorithms were compared w.r.t. execution time and energy saving. For each mapping, the energy consumed in the NoC is estimated by applying a method defined in [3]. To compare the quality of mappings, random methods were employed.

A. Experiments Description

A set of experiments with varying degree of connectivity, communication weight, NoC size, and NoC occupation was conducted in CAFES to compare the algorithms described in Section IV. A subset of relevant results is discussed here. Four sets of experiments were conducted, widely varying one given aspect while keeping others fixed or varying within a narrow range. In one set of experiments, the NoC size varied from 2x3 up to 17x17, keeping connectivity at 20%, occupation at 100% and varying communication weight randomly between 10 to 50 phits. In another, a 7x7 NoC had its communication weight varied in intervals between 1-10 phits to 1-10^4 phits. Similar choices apply to the NoC occupation experiment and the connectivity experiment sets. All algorithms, except eventually ES, were performed for each experiment set, with some results discussed next.

B. Results and Evaluation

NoC size variation is the most relevant aspect of algorithms for future SoCs, assuming good execution time and reasonable energy savings are obtained. This is indeed true for the NoC mapping problem. Considering NoC size variation, Figure 1 and Figure 2 depict the execution time and the energy gain for all approaches. Here, energy gain means the optimization of the energy consumption. All applications have 20% connectivity, random communication weight between 10-50 phits and 100% occupation. It is important to highlight that the results showed here for SA and TS are average values, obtained varying parameters like initial temperature and stop criterion.

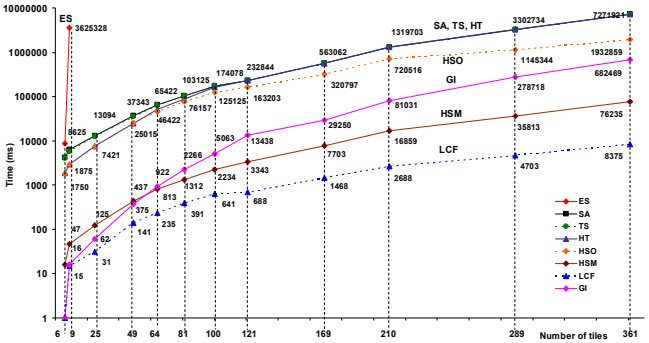

Figure 1 – Mapping execution time results

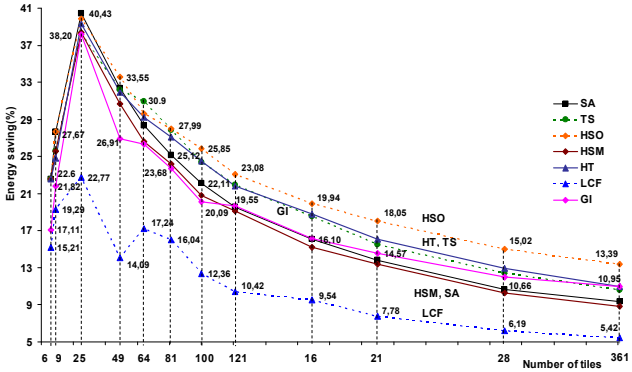

Figure 2 – Energy gain w.r.t. number of tiles compared to random mapping

Figure 3 illustrates how the connectivity of application modules affects the energy gain of mapping algorithms for a large NoC (14x15). Here, the communication weight is randomly set between 10-50 phits and the occupation is fixed at 100%.

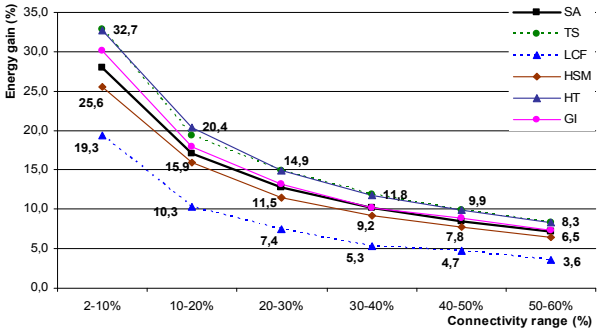

Figure 3 – Energy gain with respect to connectivity degree

Each point in the plot represents the energy gain achieved with one algorithm in each connectivity range. Clearly, increasing connectivity reduces the energy gain saving. Real application modules connect to just a small fraction of other SoC modules. Thus, the lower connectivity ranges are closest to real cases.

Concerning variation of the communication weight, Figure 4 describes the behavior of all non-exhaustive search algorithms for a small to medium NoC (7x7). Weight varies randomly within narrow and wide ranges (from [1, 10] up to [1, 10^4]). To obtain worst-case energy gain figures, connectivity is 100% (see why in Figure 3). Although this situation is unrealistic, 100% connectivity helps evaluating mapping algorithms limits, due to the needed amount of computation for calling cost functions. All algorithms but LCF obtained energy savings mostly above 10%.

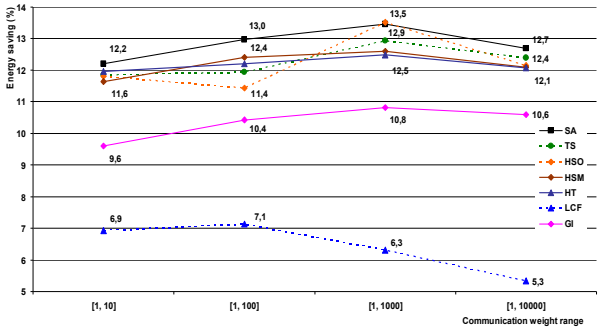

Figure 4 - Energy gain w.r.t. communication weight

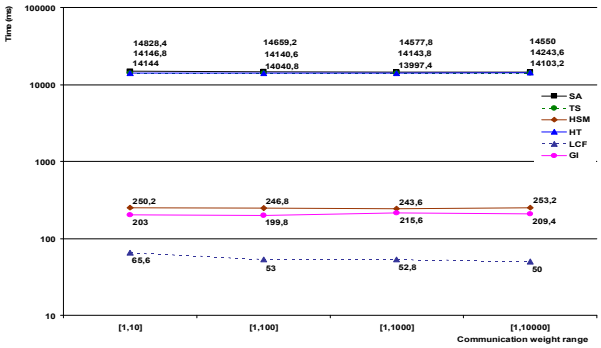

Figure 5 - Algorithm execution time w.r.t communication weight

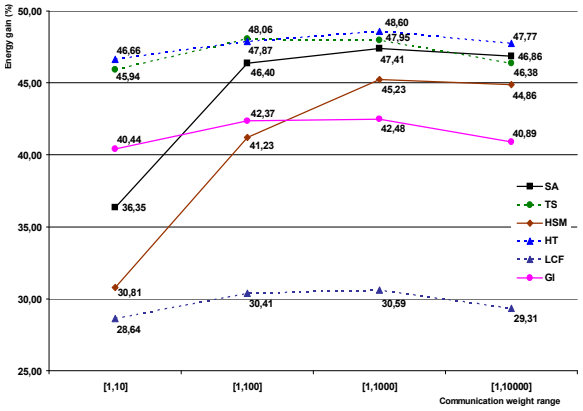

Figure 6 - Energy gain with respect to communication weight

Figure 5 and Figure 6 depict the execution time and energy savings for the same NoC with connectivity ranging from 5% to 15%. It is clear that the execution time of all algorithms depends little on the communication weight. On the other hand, the best performing mixed approach (HSM) and GI have distinct behaviors. From this result alone, it is possible to infer that HSM is better in scenarios exchanging large packets, and GI for small packet scenarios. The HSM and GI execution time are respectively 3 and 2 orders smaller than stochastic approaches.

VI. Conclusions and Future Work

A significant set of NoC mapping algorithms based on CWM targeting low energy was developed and evaluated. Exhaustive search is unfeasible for all but very small NoCs. Thus, using mapping algorithms that approximate the optimum solution is mandatory. Additionally, stochastic algorithms are very time consuming for large NoC sizes, what contributes for the adoption of heuristic solutions and mixed approaches. The GI heuristic showed good results, and LCF combined with stochastic approaches led to a good compromise between execution time and energy saving. Of all mixed approaches, combining LCF with a single-loop implementation of SA (HSM) and GI are the best ones, since they do not significantly degenerate energy gains and display execution time reductions of orders of magnitude. Further work includes extending the evaluation of the algorithms for other models such as CDM, CDCM, ACPM and ECWG [18].

References

[1] S. Kumar et al. A network on chip architecture and design methodology. In: *ISVLSI*, pp.105-112, Apr. 2002.

[2] I. Kadayif et al. Influence of communication optimizations on on-chip multi-processor energy. In: *SOC Conference*, pp.255-256, Sep. 2003.

[3] C. Marcon et al. Time and Energy Efficient Mapping of Embedded Applications onto NoCs. In: *ASP-DAC*, Jan. 2005.

[4] J. Hu and R. Marculescu. Energy- and Performance-Aware Mapping for Regular NoC Architectures. *IEEE Transactions on CAD of Integrated Circuits and Systems, v.24, n.4,* pp. 551-562, *Apr. 2005.*

[5] S. Murali and G. De Micheli. Bandwidth-Constrained Mapping of Cores onto NoC Architectures. In: *DATE*, pp. 896-901, Feb. 2004.

[6] C. Marcon et al. Exploring NoC Mapping Strategies: An Energy and Timing Aware Technique. In: *DATE*, pp. 502-7, Mar. 2005.

[7] M. Kreutz et al. Energy and Latency Evaluation of NoC Topologies. In: *ISCAS*, pp. 5866-9, 2005.

[8] G. Ascia et al. Multi-objective Mapping for Mesh-Based NoC Architectures. In: *CODES+ISSS*, pp. 182-7, 2004.

[9] W. Hung et al. Thermal-Aware IP Virtualization and Placement for NoC Architecture. In: *ICCD*, Oct. 2004.

[10] B. Sethuraman and R. Vemuri. optiMap: A Tool for Automated Generation of NoC Architectures using Multi-Port Routers for FPGAs.In: *DATE*, pp. 947-952, Mar. 2006.

[11] C-E. Rhee et al. Many-to-Many Core-Switch Mapping in 2-D Mesh NoC Architectures. In: *ICCD*. pp. 438-43, Oct. 2004.

[12] D. Wu et al. Scheduling and Mapping of Conditional Task Graph for the Synthesis of Low Power Embedded Systems. *IEE Proc. on Comp. and Digital Techniques*, v. 150, n. 5, pp. 262-73, Sept. 2003.

[13] D. Bertozzi et al. NoC Synthesis Flow for Customized Domain Specific Multiprocessor Systems-on-Chip. *IEEE Transactions on Parallel and Distributed Systems*, v. 16, n. 2, pp. 113-29, Feb, 2005.

[14] U. Ogras and R. Marculescu. Energy- and Performance-Driven NoC Communication Architecture Synthesis Using a Decomposition Approach. In: *DATE*, v. 1, pp. 352-7, 2005.

[15] J. Hu and R. Marculescu. Energy-Aware Communication and Task Scheduling for Network-on-Chip Architectures under Real-Time Constraints. In: *DATE*, pp. 234-9, Feb. 2004.

[16] S. Murali et al. Mapping and Physical Planning of NoC Architectures with QoS Guarantees. In: *ASP-DAC*, pp. 27–32, Jan. 2005.

[17] K. Srinivasan and K. Chatha. A Technique for Low Energy Mapping and Routing in NoC Architectures. In: *ISLPED*, pp. 387-392, Aug. 2005.

[18] C. Marcon et al. Modeling the Traffic Effect for Application Cores Mapping Problem onto NoCs. In: *VLSI-SoC*, 2005.