Time and Energy Efficient Mapping of Embedded Applications onto NoCs

César Marcon, André Borin, Altamiro Susin, Luigi Carro, Flávio Wagner

Instituto de Informática – UFRGS – Av. Bento Gonçalves, 9500, Porto Alegre, RS – Brazil

{marcon, borin, flavio}@inf.ufrgs.br, {susin, carro}@eletro.ufrgs.br

Abstract - This work analyzes the mapping of applications onto generic regular Networks-on-Chip (NoCs). Cores must be placed considering communication requirements, so as to minimize the overall application execution time and energy consumption. We expand previous mapping strategies by taking into consideration the dynamic behavior of the target application and thus potential contentions in the intercommunication of the cores. Experimental results for a suite of 22 benchmarks and various NoC sizes show that a 42% average reduction in the execution time of the mapped application can be obtained, together with a 21% average reduction in the total energy consumption for state-of-the-art technologies.

1. Introduction

New technologies allow many millions of transistors integrated onto a single chip and thus the implementation of complex systems-on-chip (SoC) that need special communication resources to handle very tight design requirements. In addition, deep sub-micron effects pose formidable physical design challenges for long wires and global on-chip communication. Many designers propose to change the full synchronous design paradigm to a global asynchronous and local synchronous (GALS) design paradigm [1]. GALS design subdivides the application into sub-applications. Each sub-application is a synchronous design physically placed inside a tile, and the communication between tiles is provided by an asynchronous communication resource. A network-on-chip (NoC) is an infrastructure essentially composed by a set of routers interconnected by communication channels. A NoC is suitable to deal with the GALS paradigm, since it provides asynchronous communication, high scalability, reusability, reliability, and efficient energy consumption [2].

An application composed by a set of existing cores, such as processors and memories together with their communication channels, must be mapped onto a physical network structure. To fulfill this goal, many mapping strategies have been proposed, which look for an ideal placement of the cores. For instance, in [3] and [4] a model based on a weighted graph reflecting the communication capacity of each channel is used. However, previously published approaches tend to overestimate the channel occupation, thus requiring extra bandwidth to ensure that all communications are performed within the allocated time. The overall effect is a major increase in the energy consumption. In addition, models like the ones presented in [3] and [4], which are based on weighted graph, are appropriated to model applications where the communication need is estimated in advance and not during the application execution. Hence, a conservative approach must be taken by the designer regarding bandwidth requirements, increasing the energy consumption of the NoC.

In this paper we introduce a new model, called CDM, which captures the dynamic behavior of the messages of an application. This new model allows a CAD tool to take into account the varying necessity of bandwidth along the execution of an application and hence helps reduce the total energy consumption of the system. Comparing our approach with previous published work, we achieve an average reduction of 42% in application execution time, at the same time reducing the total energy consumption of the system by 21% for state-of-the-art technologies, for a suite of 22 benchmarks and various NoC sizes.

The remaining of this paper is organized as follows. Section 2 discusses previous work related to the application-mapping problem. Section 3 describes our target architecture. Section 4 explains the mapping strategy to reduce the application execution time and the energy consumption on the target architecture. Section 5 presents experimental results, and Section 6 draws final conclusions.

2. Related Work

Hu and Marculescu [3] propose a mapping approach called *communication weighted model* (CWM), based on an *application characterization graph* (APCG), where the weight of a channel corresponds to the bit volume of the messages transmitted over this channel. With this model, they show that it is possible to reduce the energy consumption by more than 60%, when compared to ad-hoc mapping solutions.

Murali and De Micheli [4] implement a similar solution. Their CWM is also characterized by an application graph, which they call *core graph*. Their algorithm maps cores onto a mesh NoC architecture under bandwidth constraints, with the goal of minimizing the average communication delay.

Ye, Benini and De Micheli [5] propose a model to evaluate the energy consumption in a communication infrastructure considering switches, internal buffers, and interconnect wires. The same authors, in [6], describe the contention problem in NoCs and the associated performance reduction. They recommend a solution employing a routing algorithm that minimizes the energy consumption, because the required buffers in the network are reduced.

In all approaches that use the CWM strategy, essential information regarding the exact time instant at which messages are exchanged is lost.

We have developed experimental work that shows that for embedded applications and random benchmarks this information cannot be neglected. By not considering the varying nature of the communication bandwidth requirements along the execution of an application, the mapping algorithm can produce solutions that require in average 40% more execution time than a minimal solution. On the other hand, by introducing the communication bandwidth variability in the model, our tool can reduce both the application execution time and the energy consumption.

Our approach is based on a *communication dependence model* (CDM), where the application graph transports the knowledge of dependences between messages. The placement of the cores onto the NoC is based on this extra information, relating the amount of bits to be transmitted with the moment when the communication must take place. Moreover, the energy model presented in [5] is extended to consider static energy consumption, which is very relevant in new sub-micron technologies. This work assumes that application tasks are previously partitioned and assigned into a set of cores.

3. Target Architecture Description

Mapping approaches such as CDM and CWM are useful for all communication infrastructures where mappings may affect the overall performance, like hierarchical busses and NoCs. This paper approaches only NoCs as target architecture, with a 2D-mesh topology and composed by $\varphi \times \omega$ tiles. Figure 1 depicts that each tile τ contains a router r and a core c.

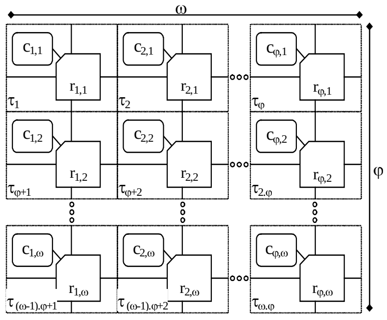

Figure 1 – Schematic of the target architecture

Tile-based architectures require the implementation of routing algorithms to transmit packets across the network. Routing algorithms can be divided into two classes: deterministic and adaptive. Deterministic routing algorithms completely specify the path from the position of the source tile to the position of the target tile. In adaptive routing algorithms, the possible paths depend on the network traffic. Adaptive routing algorithms increase the number of possible paths and require more resources because of their complexity. Therefore, we decided to choose the deterministic XY routing algorithm, which is free from deadlocks and livelocks and still route with minimal path [7]. Other works also use the same routing policy, like in [3] and [4], so that the comparison with our work may be based on the same ground rules. The algorithm behavior can be summarized in two steps: (i) first, packets are routed along the X-axis until they reach the target tile column; (ii) packets are then routed along the Y-axis until they reach the target tile row.

4. Problem Formulation

The problem of mapping application cores onto NoCs is a complex one. The designer splits the application tasks into cores. Each core has given computation and communication requirements, which can be obtained through simulation and profiling techniques. To better understand the mapping application problem, we present three definitions.

Definition 1: A *communication weighted graph* is a directed graph $CWG = <C, M>$, where $C = \{c_1, c_2, \ldots, c_n\}$ represents the set of application cores, corresponding to the set of CWG vertices. Let W_{ij} be the weight that corresponds to the bit volume of all messages exchanged between cores c_i and c_j, then the set $M = \{(c_i, c_j, W_{ij}) \mid c_i, c_j \in C\}$ symbolizes the traffic volume of all messages between all application cores. CWG has an equivalent definition to APCG [3] and core graph [4].

Definition 2: A *communication dependence graph* is a directed graph $CDG = <V, D>$. Let $v_q = (c_a, c_b, w_{ab})$ be the *q-th* message from core c_a to core c_b with bit volume w_{ab}. $V = \{v_1, v_2, \ldots, v_k\}$ denotes the set of all messages between all application cores and correspond to the set of CDG vertices, and $D = \{(v_i, v_j) \mid v_i, v_j \in V\}$ represents the set of message dependences, corresponding to the set of CDG edges. Edges are non-valued, and the edge direction means message dependence.

Definition 3: A *communication resource graph* 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, and $L = \{(\tau_i, \tau_j) \mid \tau_i, \tau_j \in \Gamma\}$ designates the set of routing paths between tiles, corresponding to the set of CRG edges. p is the total number of tiles and is equal to $\varphi \times \omega$ (Figure 1). CRG has an equivalent definition to the *architecture characterization graph* [3] and to the *NoC topology graph* [4].

The CDM approach implies the extraction of message dependence from the application cores. This means that all messages are relatively ordered by their dependences. On the other hand, the CWM approach does not take the communication ordering into consideration. Only the volume of bits exchanged between cores is considered in the CWG [3][4]. As a consequence, CWM cannot prevent contentions, thus disabling the precise estimation of the application execution time, and require a conservative approach, increasing bandwidth requirements and hence energy consumption.

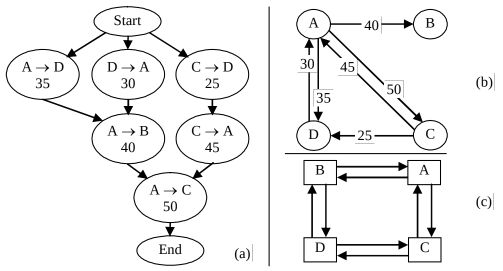

Figure 2 – CDG (a), CWG (b) and CRG (c) examples

To a better understanding of these concepts, Figure 2 depicts the CDG of a hypothetical example, where four cores C = {A, B, C, D} exchange six messages with communication rates between 25 and 50 units. Figure 2(a) shows the CDG, which highlights the message interdependences. Figure 2(b) shows the equivalent CWG, and Figure 2(c) portrays a CRG where *C* is arbitrarily mapped onto a 2x2 NoC. Since the set of CDG vertices contains information of all messages and cores, CWG can be obtained from CDG.

4.1 Energy Model

The energy consumption of the application is originated from both cores and network operation. This work focuses only on NoC energy consumption and presents a model to estimate dynamic and static energy consumption. This energy model is used as an objective function to evaluate the cost of each mapping. It is important to notice that the consumption of the cores is independent of the mapping, as this is the same assumption also found in [3][4].

Static energy consumption is mainly originated from subthreshold leakage current and is proportional to application execution time and the number of gates. Usually, static energy contributes with the smallest part of total energy consumption. However, for sub-micron technologies, the leakage current cannot be neglected, and the static energy becomes a meaningful part of total energy consumption, reaching up to 20% in state-of-the-art technologies [8]. Dynamic energy consumption is proportional to switching activity, which happens when packets move across the NoC dissipating energy inside each router and on the router interconnection wires.

Our energy model computes static and dynamic power dissipation to estimate the total energy consumption of the NoC. This work uses an approach similar to the one presented in [3] and [4] and extends the concepts to static energy consumption. We use the same concept of bit energy E_{bit} to estimate the dynamic energy consumption for each bit when the bit flips its polarity from the previous value. E_{bit} is split onto dynamic energy E_{Wbit} consumed on the switch wires, dynamic energy E_{Bbit} consumed on the buffers, dynamic energy E_{Sbit} consumed on the logic gates of each switch, and dynamic energy E_{Lbit} consumed on the links between tiles, as described in equation 1.

(1) $$E_{bit} = E_{Wbit} + E_{Bbit} + E_{Sbit} + E_{Lbit}$$

E_{Bbit}, E_{Sbit} and E_{Wbit} model the total energy consumed by a bit passing through a router. E_{Bbit} depends on the buffer size and technology to estimate how many bit flips occur to write, read, and preserve the information. When technology and routing policy are defined, E_{Bbit} and E_{Sbit} can be estimated by electrical simulation. For regular mesh NoCs, with square dimension tiles, it is reasonable to estimate that E_{Lbit} is the same for all NoC interconnections. While E_{Lbit} is directly proportional to tile dimension, E_{Wbit} becomes negligible for large tiles, since E_{Wbit} does not depend on the increase of tile size. This makes equation 2 a reasonable estimation for bit dynamic energy consumption.

(2) $$E_{bit} = E_{Bbit} + E_{Sbit} + E_{Lbit}$$

Equation 3 computes the dynamic energy consumed on the NoC by a bit traffic from core c_i to c_j, where η corresponds to the number of routers that the bit goes through.

(3) $$E_{bit}^{c_i, c_j} = \eta\,(E_{Sbit} + E_{Bbit}) + (\eta - 1)\,E_{Lbit}$$

Let λ_q be the bit volume of each message $v_q \in V$. Then,

$$E_{bit}^{v_q} = \lambda_q \times E_{bit}^{c_i, c_j} \mid (c_i, c_j) \in v_q.$$

Equation 4 gives the total amount of *NoC dynamic energy consumption* E_{DyNoC}, which considers all bit flips during the transmission of messages across the NoC.

(4) $$E_{DyNoC} = \sum_{q=0}^{k} E_{bit}^{v_q}$$

The static power consumption of each router $P_{SRouter}$ is proportional to the number of powered elements, with a very small influence of switching activity. With *p* representing the number of tiles, equation 5 computes the *NoC static power consumption* P_{StNoC}.

(5) $$P_{StNoC} = p \times P_{SRouter}$$

Static energy consumption is proportional to the total number of gates dissipating static power and to the execution time t_{exec}. Thus, equation 6 computes *NoC static energy consumption* E_{StNoC}.

(6) $$E_{StNoC} = P_{StNoC} \times t_{exec}$$

Finally, equation 7 gives the *total NoC energy consumption* E_{NoC}, which computes the consumption of static and dynamic energies.

(7) $$E_{NoC} = E_{StNoC} + E_{DyNoC}$$

With the objective of inserting the energy parameters into the CDM and CWM approaches, a NoC was described and synthesized to a 0.35micron TSMC ASIC standard cell library. The synthesis result is a netlist of cells. The manufacturer supplies energy values for the standard cell library, allowing the extraction of E_{Sbit}, E_{Bbit}, E_{Lbit}, and $P_{SRouter}$ parameters. These parameters are independent from application and NoC dimension. On the other hand, *p* and t_{exec} are application-dependent parameters. The execution time t_{exec} is measured in clock cycles, considering a 100 MHz operation frequency, and *p* is greater or equal to the number of cores in the application.

4.2 Comparing Communication Algorithms

As both *communication weighted algorithms* (CWAs) and *communication dependence algorithms* (CDAs) implement solutions for NP-complete problems [3][4], we have used a simulated annealing search method to reach the best mapping solutions in both cases. Moreover, we have also implemented exhaustive search methods, so that we could compare the quality of the solution.

For both modeling approaches the algorithms start from an initial mapping, evaluate the mapping cost, and search for a new mapping that reduces the previous cost until reaching a stop condition. CRG edges and vertices represent communication resources: links and routers, respectively. For both algorithms, cost variables are associated to each CRG edge and vertex to store the corresponding part of the mapping cost. The mapping objective function is defined as the sum of all cost variables of CRG edges and vertices. The CWA and CDA objective functions are not the same. While CWA searches only for mappings that reduce the dynamic energy consumption, as it is described in equation 4, CDA also evaluates static energy consumption, which is proportional to execution time, as it is described in equation 7. As a result, CDA indirectly searches for mappings that reduce the overall execution time.

The initial mapping of CDA or CWA is selected by randomly associating application graphs (CWG or CDG) with CRG; i.e. all cores $\in C$ are randomly mapped onto a possible tile $\in \Gamma$. To compute the mapping objective function, all cost variables of CRG edges and vertices are initially reset.

Let tiles τ_i and τ_j be mappings of cores c_a and c_b, respectively. For CWA, all bits of the communication channel (c_a, c_b), represented by W_{ab}, are associated to the correspondent cost variable of vertices and edges of CRG, starting from τ_i, following the XY routing algorithm and ending in τ_j. The cost variable of each CRG edge computes the dynamic energy of a link by multiplying W_{ab} by E_{Lbit}, and the cost variable of each CRG vertex computes the dynamic energy of a router by multiplying W_{ab} by $E_{Sbit} + E_{Bbit}$. The sum of all cost variables of CRG results in the total dynamic energy E_{DyNoC}, for a given mapping. The goal of CWA is to find mappings that reduce E_{DyNoC}. E_{StNoC} is not computed because this model is inappropriate to capture the time taken by the whole application.

While CWG considers only the communication volume, CDM captures the message dependences. Messages that have producer-consumer precedence can not be concurrent. However, temporally independent messages can occur at the same time and may consequently lead to package contention. To obtain benefits from this time notion, to each edge and vertex of CRG a cost variable list is associated, where each list position contains the energy sum of all independent messages that share the same communication resource. The algorithm considers the worst case, i.e. all independent messages that share the same communication resource produce contention. The message contention implies a larger application execution time t_{exec} and consequently more static energy dissipation E_{StNoC}. Therefore, CDA minimizes the probability of contentions by searching core mappings that spread the messages over parallel links.

The total delay of messages depends on the mapping, on the bandwidth, and on the number of bits. The algorithm computes the total delay of messages by adding the message delay only when messages are dependent from each other or when independent messages occupy the same communication resource. With the total delay of messages we apply equation 6 to obtain E_{StNoC}. Similarly to CWA, for CDA all bits of the message $v_c = (c_a, c_b, w_{ab})$, represented by w_{ab}, are associated to the correspondent vertices and edges of CRG, starting from τ_i, following the XY routing algorithm, and ending in τ_j. The cost variable list of a CRG edge computes the dynamic energy of a link, in a given period, by multiplying w_{ab} by E_{Lbit}. The cost variable list of a CRG vertex computes the dynamic energy of a router, in a given time period, by multiplying w_{ab} by $E_{Sbit} + E_{Bbit}$. The sum of all cost variables of CRG results in the total dynamic energy E_{DyNoC}, for a given mapping. CDA uses equation 7 as an objective function to evaluate the mapping cost. The goal of the CDM algorithm is to find mappings that minimize E_{NoC}.

If the mapping cost achieved with a new mapping is smaller than the one previously stored, the current mapping and cost are saved for further comparison. Simulated annealing may accept worse mappings, depending on the temperature, which is a convergence parameter of the algorithm. While the stop condition has not yet been reached, a new mapping is randomly chosen, and the cost is evaluated again. While simulated annealing considers parameters as initial temperature and number of iterations, the stop condition for an exhaustive search requires the evaluation of all mappings.

In embedded applications like the graphical ones used in this work, the number of messages between cores is much larger than the number of cores. Since each vertex of CDG represents a message between two cores and each vertex of CWG represents a core, CDGs are larger than CWGs, implying more CPU time and more data storage area for the algorithm execution. A comparison between CDA and CWA is presented in Section 5.

4.3 Comparing Communication Models

The main advantages of CWM are (*i*) easy extraction of the application core graph (CWG), since this can be done by simulation techniques; (*ii*) low computational complexity; and (*iii*) the accurate estimation of E_{DyNoC}, since dynamic energy may be well computed by the total bit traffic in the NoC. On the other hand, the extraction of CDG is hard to be automatically obtained, since simulation allows the extraction of the possible message ordering, but not the message dependences. This implies that CDGs have to be described in design time by hand, and this is an error prone task. The greater complexity of CDM directly reflects in the complexity of the algorithm to deal with it, which increases the computation time and the memory usage. However, CDM captures both the bit volume, which allows computing the value of E_{DyNoC}, and the message ordering, which allows estimating the instants of time when more than one message can pass through the same link, and consequently avoiding such occurrence by a better core mapping. Such approach is pessimistic, since not all communications that can occur concurrently will happen concurrently. Even so, the overall application performance tends to increase, if potential contentions were avoided.

The global communication behavior of a certain application can be expressed as a function of start of transmission times, bit volume, and transmission rate of each message. For many applications, the exact determination of communication needs at design time may not be possible, since it depends on the specific input data that can only be available at runtime. These data have an important influence on the bit volume of messages and less or no influence on other parameters. The order in which messages are transmitted is usually not changed, since it depends on the algorithm executed by the application, which is usually fixed in embedded systems. Since CDM models the dependency between messages, a feature not available in CWM, it allows evaluating the potential for contention among dependent messages, even without the precise knowledge of the exact bit volume for each message. This capacity enables to find mappings that further reduce energy consumption and execution time. Simultaneously, it makes CDM less sensitive to input data variations at runtime, as it will be shown in the next section.

4.4 Problem Illustration

This section illustrates the application of the CDM and CWM approaches with the same hypothetical example of Figure 2, where two mappings imply different execution times and energy consumptions. It is also shown that CWM is not suitable to capture such differences, since this model computes the same energy consumption for both mappings.

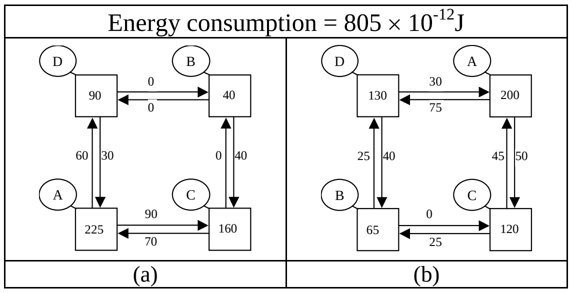

Figure 3 – Two mappings with energy estimated by CWA

Figure 3 illustrates two mappings of the example shown in Figure 2. Just for illustration purposes, this example assumes that $E_{Sbit} + E_{Bbit} = E_{Lbit} = 1.\ 10^{-12}$ J. Each vertex and edge of CRG is annotated with its total amount of energy consumption. As CWM cannot capture contention problems, CWA estimates that both mappings consume the same energy (805×10^{-12} J).

Figure 4 shows the same mappings of Figure 3, now evaluated with CDM. Each edge and each input link of each vertex is annotated with its energy at a given slice of time. For instance, in Figure 4 (a) the tile corresponding to core D is annotated with 35_{S1} and 25_{S1}, which means that there are 2 messages with 35 and 25 bits, respectively. These two messages are concurrent – see Figure 2(a) – and are annotated in the cost variable list of vertex D. Both use the SOUTH (S) link of core D (one from A to D and the other from C to D) increasing the overall execution time of the application and, consequently, the static energy E_{StNoC}. Just for illustration purposes, consider t the necessary number of clock cycles to transfer one bit from one tile to its neighbor tile and $P_{StNoC} = 1\ 10^{-12}$ J/t the power consumed by clock cycle. In this case, the energy consumption and execution time are 2.7% and 20% greater, respectively, when comparing mapping (a) with mapping (b). These differences are only captured with CDA. We emphasize that when the number of messages and cores of the application increases these differences also increase, as it will be shown in Section 5. The overhead in performance requirements of CWA-like approaches is the cause of extra power dissipation. In the experiments of the next section we show that the CDM approach can remove this overhead.

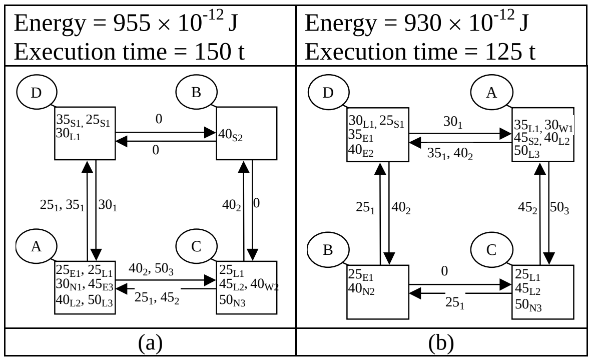

Figure 4 – Mappings of the Figure 3 estimated by CDM

5. Experimental Results

Table 1 summarizes the characteristics of 22 applications mapped onto 8 different NoC sizes (NS). There are 4 embedded applications (a distributed Romberg integration [9], an 8-point Fast Fourier Transform [10], and 2 image applications for object recognition and image encoding) with some variations, in a total of 8 embedded applications. The remaining applications are benchmarks randomly generated by a proprietary system, which is similar to TGFF [11]; however, our system describes a benchmark by a CDG, which represents message dependence and bit volume of each message. The chosen application characteristics are: number of cores (NC), number of messages between cores (NM), and total amount of bit traffic during application execution (TBT). NS is equivalent to the number of CRG vertices, NC corresponds to the number of CWG vertices, and NM matches the number of CDG vertices.

Table 1 – NoC dimensions and application characteristics

	NS	NC	NM	TBT
Random benchmarks	2 x 2	3; 4; 4	15; 12; 23	213; 450; 23,234
	3 x 2	5; 6; 6	43; 17; 43	78,817; 174; 49,003
	2 x 4	7	33	23,235
	3 x 3	7; 9; 9	16; 18; 32	1,600; 1,860; 43,120
	2 x 5	8; 9; 10	24; 51; 22	2,215; 23,244; 322,221
	3 x 4	11	62	123,337
Embedded applications	3 x 2	5	16	1,600
	2 x 4	5; 8	16; 18	1,600; 5,930
	3 x 3	8	31	4,655,025
	3 x 4	10; 12	15; 25	3,100; 2,578,920
	8 x 8	62	344	9,799,200
	10 x 10	93	415	562,565,990

For each application, the best mapping achieved with CWM is compared to the best mapping achieved with CDM. Gains obtained with CDM when compared to CWM are summarized in Table 2. ES represents evaluations obtained by exhaustive search, while SA symbolizes evaluations obtained with simulated annealing algorithm. ETR gives the average execution time reduction, and ECS denotes the average energy consumption saving, for a given technology, when the best mapping obtained with CDM is compared to the best one obtained with CWM. $ECS_{0.35}$ column refers to values obtained from 0.35micron technology, and $ECS_{0.07}$ column refers to values obtained by scaling results from 0.35micron to 0.07micron [8].

Table 2 – Average energy consumption saving and execution time reduction obtained from comparison of CWM and CDM evaluations

	Algorithm	NS	ETR	$ECS_{0.07}$	$ECS_{0.35}$
Random benchmarks	ES / SA	2 x 2	32 %	16 %	0,51 %
		3 x 2	39 %	17 %	0,54 %
		2 x 4	31 %	15 %	0,48 %
		3 x 3	43 %	25 %	0,8 %
		2 x 5	50 %	27 %	0,86 %
		3 x 4	47 %	25 %	0,8 %
Embedded applications	ES / SA	3 x 2	39 %	17 %	0,54 %
		2 x 4	31 %	15 %	0,48 %
		3 x 3	43 %	25 %	0,8 %
		3 x 4	47 %	25 %	0,8 %
	SA	8 x 8	44 %	22 %	0,7 %
		10 x 10	51 %	28 %	0,89 %
	Total average		42 %	21 %	0,67 %

As seen in the ETR column, CDM results, in average, a reduction of 42% of execution time when compared to CWM. The $ECS_{0.35}$ column illustrates a very small energy consumption saving, since the static leakage current is not that important for this technology generation. However, for sub-micron technologies, where the static dissipation is more relevant, there is a significant reduction in energy consumption (21% in average), as we can see in column $ECS_{0.07}$. In addition, Table 2 shows a slight tendency of better energy consumption savings and execution time reduction when the NoC size increases. Finally, results obtained with exhaustive search are very similar to the ones achieved with simulated annealing. For all small NoCs (up to 3x4 or 2x5), both algorithms reached the same results. For larger ones (8x8 and 10x10), it is not possible to find optimal mappings with the exhaustive search within a reasonable computation time.

The mapping cost evaluation of CWA considers mainly the number of links between cores. At the same time, the number of messages has higher influence in CDA, because messages cannot occupy the same link at the same time. This leads the CWA computational complexity to be proportional to the number of links (NL) and the CDA computational complexity to be proportional to the number of messages (NM). In embedded applications, NM may be much larger than NL. However, the increase in CPU time with the increase of the NM/NL ratio is practically linear and has a small slope. In our experiments, the worst case of CDA took only 15% more CPU time then CWA.

The main drawback of CDA is associated to the extra memory to run the algorithm, since for CDA all vertex and edges of CRG preserve a list of concurrent messages, while CWA implies the use of only one data element for each vertex and edge. In our experiments, the worst case of CDA took 26 times more memory than CWA.

6. Conclusions

This paper addressed the problem of mapping application cores onto NoCs. A communication dependence model (CDM) is introduced and compared to a communication weighted model (CWM). We conclude that a mapping algorithm that implements CDM is able to reduce some application requirements, when compared to a mapping algorithm that implements CWM. Experimental results show an average reduction of 42% in the application execution time. The CDM approach also reduces the energy consumption. For instance, for a 0.07micron technology an average of 21% in energy savings is achieved. This reduction is obtained because CDM may avoid or, at least, reduce message contention, while CWM may not. Moreover, to map applications where the communication needs of each core are not known at design time, CDA may also achieve mappings that reduce the energy consumption and execution time, while CWA may not. Algorithms that implement CDM present only a moderate increase in the execution time when compared to algorithms that implement CWM, with much better mapping results.

References

[1] A. Iyer and D. Marculescu. *Power and Performance Evaluation of Globally Asynchronous Locally Synchronous Processors*. **ISCA**, pp. 158-168, May 2002.

[2] W. Dally and B. Towles. *Route Packets, Not Wires: On-Chip Interconnection Networks*. **DAC**, pp. 648-689, June 2001.

[3] J. Hu and R. Marculescu. Energy-Aware Mapping for Tile-based NoC Architectures Under Performance Constraints. **ASP-DAC**, pp. 233-239, January 2003.

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

[5] T. Ye; L. Benini and G. De Micheli. *Analysis of Power Consumption on Switch Fabrics in Network Routers*. **DAC**, pp. 524-529, June 2002.

[6] T. Ye; L. Benini and G. De Micheli. *Packetization and routing analysis of on-chip multiprocessor networks*. **Journal of Systems Architecture**, vol. 50, issues 2-3, pp. 81-104, February 2004.

[7] C. Glass and L. Ni, *The Turn Model for Adaptive Routing*. **ISCA**, pp. 278-287, May 1992.

[8] D. Duarte, N. Vijaykrishnan, M. Irwin, H-S Kim, G. McFarland. *Impact of Scaling on The Effectiveness of Dynamic Power Reduction Schemes*. **ICCD**, pp. 382-387, September 2002.

[9] R. Burden and J. D. Faires. **Study Guide for Numerical Analysis**, McGraw-Hill, New-York, 2001.

[10] M. Quinn. **Parallel Computing- Theory and Practice**, McGraw-Hill, New-York, 1994.

[11] R. Dick, D. Rhodes and W. Wolf. *TGFF: task graphs for free*. **CODES/CASHE**. pp.97–101, March 1998.