Energy and Latency Evaluation of NoC Topologies

Marcio Kreutz[1], Cesar Marcon[1,2], Luigi Carro[1], Ney Calazans[2] and Altamiro A. Susin[1]
GME – Informática – UFRGS[1] - Porto Alegre, RS, Brazil – PUCRS[2] - Porto Alegre, RS, Brazil
[kreutz, marcon,carro]@inf.ufrgs.br, calazans@inf.pucrs.br, susin@eletro.ufrgs.br

Abstract — **Mapping applications onto different networks-on-chip (NoCs) topologies is done by mapping processing cores on local ports of routers considering requirements like latency and energy consumption. In this work, an algorithm devoted to evaluate different topologies is proposed. The evaluation starts with an application model called Application Communication Pattern (ACP), which specifies tasks with the computation load and communication profile. ACP focuses on communication aspects and is an appropriate model to obtain mappings that comply with application requirements. ACP allows fast analysis over many NoC topologies, helping the system designer to evaluate the communication performance of a NoC-based system; this performance strongly depends on the placement of the cores, and it is computationally hard to find the optimal placement.**

I. INTRODUCTION

Future billion transistors Systems-on-Chip (SoCs) will allow the development of new applications, which will work in a distributed way and require reusable communication architectures offering scalable bandwidth and parallelism. Networks-on-Chip (NoCs) emerge as a potential tile-based architecture to meet such requirements. NoCs are communication infrastructures composed by a set of routers interconnected by communication channels, which can provide asynchronous communication between synchronous domains. An important issue for NoC-based system designers will be to find a solution of the communications-to-network mapping problem in order to satisfy the communication requirements.

In the upcoming years, a NoC is expected to accommodate more than 10 x 10 tiles [1]. The search for appropriate models and algorithms for mappings problems becomes mandatory. The efficient implementation of tile-based architectures requires efficient mapping strategies. This paper introduces the *Application Communication Pattern* (*ACP*), a model that enables to capture not only the communication capacity, but also the communication ordering. We use ACP to evaluate latency and energy consumption on different NoC topologies. The goal is to find, among regular and irregular NoC topologies, the one that better fits the application requirements. This will help on the design space exploration for NoCs at earlier stages of design.

II. RELATED WORK

Hu and Marculescu [2] showed that mapping algorithms reduce over 60% of energy consumption when compared to ad hoc mapping solutions. Murali and De Micheli, in [3], implement a similar solution. The focus of their papers is to present an algorithm that maps the cores onto mesh NoC architecture under bandwidth constraints, aiming to minimize the energy consumption and the average communication delay. We emphasize that application models, as the one presented in [2][3], omit essential information to estimate the latency of the application, since these models do not consider precisely the time where each communication take place. The knowledge of the communication *ordering* leads to the ACP model (presented in this paper), which aims for a better mapping solution with low extra computational effort, if compared to previous models.

Murali and De Micheli [4] extend the work presented in [3], by the introduction of a tool called SUNMAP. SUNMAP built inside a predefined library of topologies and uses a multi-objective function, which encompasses average communication delay, area and energy consumption. The main objective of the tool is to select automatically the best topology for a given application and to generate a mapping of cores onto that topology.

Hu and Marculescu [5] introduce a model that captures communication and computation scheduling, which is represented by communication task graph (CTG). CTG allows obtaining more accurate results than the one presented in [2][3], since it takes into account the effects of the traffic dynamics. However, while the input data of ACP is easily extracted through the application simulation, CTG implies an extra effort, since the designer has to describe the application and also its computation and communication scheduling.

Our approach uses ACP, which models the messages ordering and traffic load. Similarly to [4], this paper explores the design space for NoC topologies, however, we employ analytical energy models in the search for an optimized topology. In addition, we extend the work with the analysis of the effects of the tile size in the energy consumption.

III. PROBLEM FORMULATION

Given a distributed application, we can state our mapping problem as the task of minimizing the latency and energy consumption by determining the better place for cores on different NoC topologies. The solution for this problem relies in a very complex task, because more than one variable is considered concurrently in the search space: core place, topology and tile size.

It is assumed an application whose tasks were previously mapped onto a set of cores, so we envision a scenario where functions are distributed among a number of cores in a SoC.

We have defined an *application communication pattern* (ACP) to represent the semantics of communicating cores. Communications can be mainly described by the relationships among the cores, by the parallelism among messages exchanges and by the order in which the communications must occur. To model such behavior, it is possible to define a data structure like a “list of sets”, where each element of a “time tag list” points to a set of messages.

Definition 1: An ACP is a list of sets. Let $C = \{c_1, c_2, \ldots, c_n\}$ be the set of application cores, and $b_q \in \mathbb{N}$ the number of bits of the q-th message. Then $m_{ijq} = (c_i, c_j, b_{ijq}) \mid c_i, c_j \in C$ is the q-th message from core c_i to core c_j with b_{ijq} bits. Let $M = \{m_{ijq} \mid c_i, c_j \in C\}$ be the set of all messages between application cores and $\boldsymbol{m}$ be a subset of M. ACP = $\{(t, \boldsymbol{m}) \mid t \in \mathbb{N}, \ \boldsymbol{m} \subset M\}$ represents an ordered list of message sets, such that t is a time tag that marks the start time of all messagens of $\boldsymbol{m}$.

The communication architectures can also be modeled as a graph, whose vertices represent tiles and the set of oriented edges express all the links given by the network topology. This data structure is defined as the *communication resource graph* (CRG).

Definition 2: A $CRG = <\Gamma, L>$ is a directed graph, 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 links from τ_i to τ_j, corresponding to the set of CRG edges. The way the edges are connected represents the network topology. Each directed edge $l_{ij} = (\tau_i, \tau_j)$ has associated a structure s_{ij} composed by parameters that expresses the link characteristics in a given topology, such as a link width.

Figure 1 illustrates the above definitions through a hypothetical application with four cores $C = \{a, b, c, d\}$, six messages, and a 2x2 NoC. Figure 1(a) depicts an ACP = $\{(t_1, \{(a, b, 15), (c, a, 20)\}), (t_2, \{(c, a, 15), (a, d, 15), (b, d, 40)\}), (t_3, \{(d, b, 15)\})\}$. Figure 1(b) depicts a CRG with an arbitrary valid mapping of C onto CRG, generating the following association: $\{(\tau_1, b), (\tau_2, a), (\tau_3, d), (\tau_4, c)\}$.

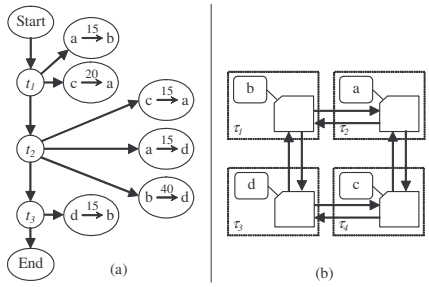


Figure 1 – ACP (a) and CRG (b) examples

For each application message, it is necessary to find in the CRG a path between its sender and receiver vertices in order to determine if the bandwidth offered by this path matches that one required by the application.

Definition 3: A path $p_{ij} = (\tau_i, l_{i\gamma}, \ldots, \tau_w, l_{wj}, \tau_j)$ is an alternating sequence of CRG vertices and edges, to transport a message from core c_i to core c_j. A path is formed according to the routing strategy implemented in the network tiles the CRG represent. p_{ij} may also correspond to an indirect path in an irregular topology, for instance, a path that uses a router of level two in a fat-tree topology.

Finally, in order to find a valid mapping, one must map each message of the ACP to links and local ports associated to tiles of the CRG.

Definition 4: Given a CRG, for each $m_{ijq} \in$ ACP there exists a corresponding $p_{ij} \in$ CRG, i.e. there is a mapping function F: ACP $\rightarrow$ CRG such that $\forall\, m_{ijq} \in$ ACP $\exists\, p_{ij} \in$ CRG.

A. *Energy Model*

This work uses an energy model similar to the ones presented in [2][3], and extends those concepts to estimate energy for different topologies. We use the same concept of bit energy *Ebit* to estimate the dynamic energy consumption for each bit. *Ebit* is split into: bit dynamic energy consumed on the buffers (*EBbit*), on the logic gates of each switch (*ESbit*); and on the links between tiles (*ELbit*), which is directly proportional to tile dimension. Because of the strong topological dependence to compute *ESbit*, *EBbit* and *ELbit*, we define α as a function to estimate the total sum of *ESbit* and *EBbit* and φ to estimate *ELbit*. Equation 1 computes the dynamic energy consumed by a single bit traversing the NoC, from tile τ_i to tile τ_j - where communicating cores were previously mapped - and *TOP* represents the topology selected for evaluation.

$$Ebit_{ij\ TOP} = \alpha_{TOP}(i, j, EBbit, ESbit) + \varphi_{TOP}(i, j, ELbit) \quad (1)$$

To estimate mesh, folded torus and fat-tree NoC topologies, we assume some considerations for all topologies: *(i)* the router area is insignificant front of the core area; and *(ii)* all tiles are regular and with the same square dimensions. These considerations allow estimating the energy consumption for each topology, permitting a comparative evaluation of each layout at earlier design stages.

Assuming *(ii)*, Figure 2(a) depicts that vertical and horizontal links have the same size, i.e. $l_{mV} = l_{mH} = l$. Since *ELbit* is proportional to l size, for mesh topology function φ of equation 1 can be approximated to equation 2, where η corresponds to the number of routers through which the bit passes from tile i to tile j.

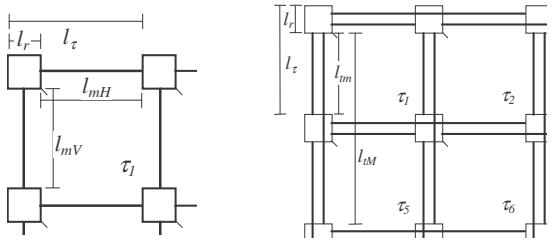
$$\varphi_{MESH}(i, j, ELbit) = (\eta\text{-}1)\ ELbit \quad (2)$$

Assuming *(ii)* Figure 2(b) depicts that $l_{tm} = l$, and as a result of *(i)* if $l_r << l_\tau$, than $l_{tM} \cong 2l$ making equation 3 an approximation of function φ for folded torus topology.

$$\varphi_{TORUS}(i, j, ELbit) = (2 \text{ or } 1) \times (\eta\text{-}1)\ ELbit \quad (3)$$

ESbit and *EBbit* of mesh and folded torus topologies are similarly estimated since routers are the same. Then, the function α of equation 1 can be approximated to equation 4.

$$\alpha_{MESH\_OR\_TORUS}(i, j, EBbit, ESbit) = \eta\,(ESbit + EBbit) \quad (4)$$



Mesh topology (a) Folded torus topology (b)

Figure 2 – Partial view of NoCs

The irregularity of fat-tree topology implies paths with different sizes and consequently different *ELbit*. For instance, Figure 3 shows the minimum paths size among one router of level one and all routers of level-two of a fat-tree topology with sixteen tiles. Considering the depicted routing layout and assuming (i) and (ii) $la \cong l/2$, $lb \cong l$, $lc \cong l$ and $ld \cong 2l$. For instance, *ELbit* is proportional to 3l (2 *la* and 2 *lb*), when the path uses a router of level one in the same tile of the router of level two. On the other hand, *ELbit* is negligible, if the routing does not use routers of level two. The function α of equation 1 can be approximated to equation 5 for fat-tree topology.

$$\alpha_{TREE}(i, j, EBbit, ESbit) = (0, 3, 4 \text{ or } 5)\ ELbit \quad (5)$$

The routing inside a level one fat-tree router implies the use of a buffer and a switch that consumes – in average – double energy than the switches of mesh or folded torus topologies. This happen

because switches and buffers of routers from level 1 must provide a path for packets coming from the upper level as well a form the same router. When a router of level two is used, there are three buffering and switching stages: one for each level, and for the target. In this case, equation 6 represents *ESbit* and *EBbit* computation.

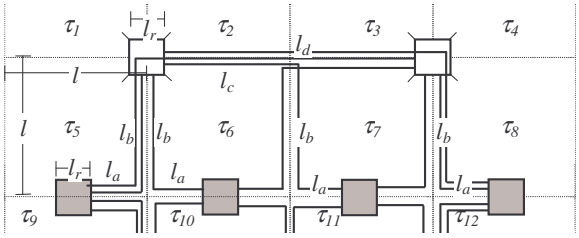


Figure 3 – Partial view of a NoC with fat-tree topology. It is depicted 2 level one and 4 level two routers (shaded)

$$\alpha_{TREE}(i, j, EBbit, ESbit) = (1 \text{ or } 3)\ ESbit + (2 \text{ or } 4)\ EBbit \quad (6)$$

The complete $Ebit_{ij}$ can be estimated by equations 7, 8 and 9, for mesh, folded torus and fat-tree topologies, respectively.

$$Ebit_{ij\ MESH} = \eta\ (ESbit + EBbit) + (\eta\text{-}1) \quad (7)$$

$$Ebit_{ij\ TORUS} = \eta\ (ESbit + EBbit) + (2 \text{ or } 1)\ (\eta\text{-}1)\ ELbit \quad (8)$$

$$Ebit_{ijTREE} = (1 \text{ or } 3)\ ESbit + (2 \text{ or } 4)\ EBbit + (0, 3, 4 \text{ or } 5)\ ELbit \quad (9)$$

Let n_{ijq} be the total amount of bits of a message $m_{ijq} \in M$ going from c_i to c_j. Then, for all topologies, $Ebit_q = n_{ijq} \times Ebit_{ij}$. Equation 10 gives the total amount of *NoC dynamic energy consumption* (*EDyNoC*), which computes all bit traffic of all k messages.

$$EDyNoC = \sum_{q=0}^{k} Ebit_q \quad (10)$$

Given this energy model, an evaluation algorithm can be developed to optimize NoC topologies regarding energy consumption. In the next section our strategy to optimization and the associated algorithm are detailed.

IV. Evaluating Mapping onto NoC Topologies

The problem of mapping cores onto NoC topologies is an instance of the *set covering problem* (SCP) which is proven to be NP-complete [9]. The search space grows exponentially with the system size: even for a 4x4 NoC there are 16! possible mappings. To solve the problem we adopt the *Tabu Search* (TS) algorithm [7], since it has been used to solve many covering problems in many different areas [8]. Figure 4 depicts the pseudo-code for the TS algorithm, where t stands for one step of the algorithm, k is the cost function for energy consumption and latency, Y is the set of all possible solutions and nt is the total number of iterations the algorithm should execute.

The general strategy of TS is to explore, from a set of p resources $S = (1, 2, \ldots, p)$, all possible moves from the current solution to a neighboring one. The move leading to a neighboring solution can be accepted, even if this results in a deterioration of the objective function. To prevent the search process from cycling, a Tabu list (T) is used to store the last moves for a certain number of iterations (nt). Thus, all solutions, which can be obtained by applying a move stored in T, are excluded from the search. An aspiration criterion allows overriding the Tabu status of a move, for instance, if the move leads to a new best solution. The OPTIMUM function selects a best move among a set of neighbors solutions.

```
Tabu_Search(resources S)  {
   select an initial solution:
       y ∈ Y e y* = y; T = ∅
1  if (S(y)-T) == ∅
       stop;
   else
       t = t + 1;
   select best y = OPTIMUM(s(y): s ∈ S(y)-T);
   if k(y) < k(y*)  // y* → best solution
      y* = y;
   if t > nt
     stop;
   else
      update T;
   go to 1;
}
```

Figure 4 – Pseudo code for a Tabu Search algorithm

The S set is made equal to cores places, where each new solution (a move) in the search space means a core swapping, which changes the source and destination for communications; $S = \{c_2, c_4, c_8, \ldots, c_n\}$; where $S[0] = c_2$ means that core c_2 is placed on the tile *0*.

The values for latency and energy consumption may vary for each application execution, due to network *contentions*. Due to its unpredictable behavior, contentions can only be obtained at execution time. We developed a simulator that simulates C++ models for generic components of NoC routers. By simulating a CRG for a target NoC, a system designer can observe how NoC components behave when they send and receive application messages. The NoC simulator is used to evaluate each mapping given by each step in the TS algorithm.

A *simulation step* stands for all messages sent concurrently by the application at each instant of time t, according to the Definition 1. Since the TS algorithm simulates the communications for each solution found by the OPTIMUM function, it was necessary to develop a simulator, which could run as fast as possible. This is the reason because all NoC components were described in C++, accompanying with a dedicated simulator.

We have modeled direct (torus and mesh) and indirect (fat-tree) topologies for evaluation. For direct topologies, it was employed the handshake flow control, a deterministic and source-based routing strategy (XY routing), wormhole packet switching, round-robin arbitration and input buffering. The fat-tree NoC differs in the routing policy, which is partially adaptive – to send messages to the second level of the tree structure – and partially static, the address of the destination node is conceived before a message is sent. These settings are commonly used for NoCs, so they can represent typical NoCs considered for SoCs design.

V. Experimental Results

To corroborate the efficiency of the proposed TS-based algorithm we have tested three applications running on three different NoC topologies: a 4x4 mesh, a 4x4 folded torus and a fat-tree with 16 terminals. Communication channels width was set to 32 bits and the length of messages vary stochastically between 100 and 800 bytes. The first application is a mathematical application that calculates the *Romberg* integration [9]; the second one is an 8-point Fast Fourier Transform (*FFT*) [10]; the third one is an "*image* processing" application for object recognition in a video frame.

The results for latency and energy Tabu-based optimization are given in Figure 5 and Figure 6, respectively. Figure 5 shows that the folded torus and fat-tree topologies could be better optimized for all applications due to higher level of parallelism offered in comparison to the mesh topology. The sequential accesses to a memory and to a central processor in the image application increase the latency of folded torus topology when compared to fat-tree topology. Its reduced number of hops justifies the best result of fat-

tree topology.

When energy consumption is taken into account, the mesh structure presented best results, as outlined in Figure 6. The reason for that relies on the shorter wires length a mesh structure has, in comparison to others. For the image application, an optimal result could be achieved for the fat-tree topology, because the energy consumption is nearly as good as the one achieved for the mesh. The reason for that relies on the fact that for this structure the TS algorithm could find the best local minima, which implies in lower switching activities, as well as lower usage of the longer wires.

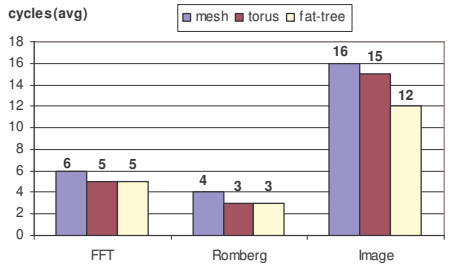


Figure 5 – Latency Optimization

We conclude that, by taking an intersection between latency and energy consumption results, the fat-tree is the better choice for executing the image application.

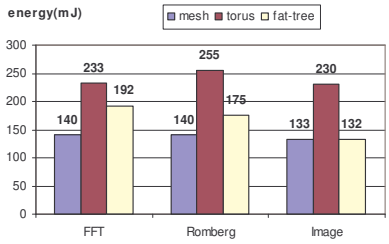


Figure 6 – Energy Optimization

Although the folded torus topology has presented good results regarding latency for all applications, it is not a good choice when energy consumption is critical. A good trade-off between latency and energy consumption could be found for the mesh topology, because it has – in average – about 10% lower performance, but 35% lower energy consumption. Another interesting experiment regards on tiles area overhead. The area of a tile can be enlarged to accommodate the area of the target core. This may increase the length (and capacity load) of communication wires used to connect them, which lead to new values for *ELbit* (defined in section III.A). In Figure 7, it is outlined the results achieved by running the Romberg application for tiles 2, 4 and 8 times larger than the ones previously tested.

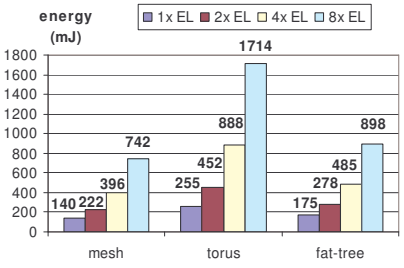


Figure 7 – Latency optimization for different tiles size

As previously shown in Figure 6, the mesh topology can achieve the best results for energy consumption, regardless of the area of tile it is mapped to. Another interesting point concerns the fact that all topologies tested, suffer *proportionally* the same penalty – in energy consumption – for longer wires in communication channels: the bigger *ELbit* is, the higher is the energy consumption at the same proportion for all topologies. Therefore, if bigger cores replace the original ones, the NoC optimized and selected to run the application will remain the same.

Finally, regarding simulation performance, for all Tabu executions, it needed not more than two or three seconds to find an optimal solution in a system composed by a 1 GHz Pentium IV machine with 512 MHz of memory. Therefore, the simulator allows larger applications to be explored in affordable time.

VI. Conclusions and Future Work

In this work, we have presented an approach to evaluate NoC topologies, which is based on a simulation engine and on a Tabu-based heuristic optimization algorithm. Since NoCs architectural components exploration turn to a huge design space to be inspected, we have chosen to employ a heuristic algorithm to evaluate different network topologies. This was made with the objective of network concurrency exploration, in order to find a structure that could better optimize latency and energy consumption. To do that, an analytical energy model was developed, which is able to capture energy consumption estimations for regular and irregular topologies. The design exploration proposed encompasses NoC concurrency evaluation by simulating applications messages on different topologies and by mapping processor cores in a way that reduces NoC contentions.

We showed that the algorithm employed can find optimal solutions for regular and irregular topologies in an affordable time. In addition, this paper shows that fat-tree topology is a strong candidate to fulfill the latency constraints for many applications, while mesh topology achieves the less energy consumption, and for a image processing application the best trade-off between latency and energy consumption is obtained with fat-tree topology.

As future works, we plan to continuously improve the heuristic by adding processor cores execution times in order to evaluate not only the communication subsystem, but also a complete SoC.

References

[1] S. Kumar et al. *A Network on Chip Architecture and Design Methodology*. **IEEE Computer Society Annual Symposium on VLSI**, pp.105-112, April 2002.

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

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

[4] S. Murali and G. De Micheli. SUNMAP: A Tool for Automatic Topology Selection and Generation for NoCs. **DAC**, pp.914-919, June 2004.

[5] J. Hu and R. Marculescu. Energy-aware communication and task scheduling for network-on-chip architectures under real-time constraints. **DATE**, pp.234-239, February 2004.

[6] M.R. Garey and D.S. Johnson. Computers and Intractability: A Guide to the Theory of NP-Completeness. **W.H. Freeman and Co**. 1979.

[7] F. Glover. *Tabu search – Part I*. **ORSA Journal on Computing**. vol. 1, pp.190-206, 1989.

[8] F. Glover. *Tabu search – Part II*. **ORSA Journal on Computing**. vol. 1, pp.4-32, 1990.

[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.