THE IMPLICATIONS OF REAL-TIME BEHAVIOR IN NETWORKS-ON-CHIP ARCHITECTURES

Edgard de Faria Corrêa[1,2], Eduardo Wisnieski Basso[1], Gustavo Reis Wilke[1], Flávio Rech Wagner[1], Luigi Carro[1]
[1] *Instituto de Informática-Universidade Federal do Rio Grande do Sul – Porto Alegre-Brasil;*
[2]*Superintendência Informática-Universidade Federal do Rio Grande do Norte – Natal-Brasil*

Abstract: This work discusses the adaptation of NoCs to real-time requirements, in particular with respect to the fulfillment of task deadlines. It is shown that, for soft real-time systems, the number of missed deadlines can be substantially reduced by the utilization of a routing mechanism based on message priorities. A core placement strategy based on message bandwidth requirements and also on message priorities can also reduce the number of missed deadlines. The paper also discusses the impact of these strategies on the energy consumption of the system and shows that an interesting design space can be explored.

Key words: Systems-on-Chip, SoC, Networks-on-Chip, NoC, Real-Time.

1. INTRODUCTION

Technology scaling improvements will allow SoCs with dozens of IP blocks and large amounts of embedded memory until the end of this decade[1]. IPs can be for instance CPU or DSP cores and video stream processors, allowing new applications in the fields of telecommunication, entertainment, and consumer electronics. Communication architectures providing from dozens to hundreds of Gbit/s (or Tbit/s)[2] will be required. Furthermore, these architectures must follow reusable templates, in order to amortize design costs among several designs and to meet time-to-market pressures[3].

The typical reusable communication template in current SoCs is based on the bus approach. But, this approach has strong constraints that will inhibit

its use in future SoCs. First, a bus does not scale with the system size, and its bandwidth is shared by all the cores attached to it. Second, its operating frequency degrades with the increase in the number of cores, because of the growing capacitive loading in the wires. Finally, the power consumption increases with the wire length, which increases with the circuit size.

Recent works[1,2,4] have proposed the use of interconnection networks[5] as an alternative approach to interconnect cores in SoCs. The overall idea is that such NoCs will meet the major requirements for future SoCs: reusability, scalability, and parallelism, while coping with power constraints and clock distribution. Furthermore, the application requirements, such as time restrictions and performance, can be met with the correct customization of NoC features, like routing policy, speed of the router, and buffer size.

This work discusses the application of NoCs to real-time (RT) systems, in particular regarding task deadlines. Since predictability of execution and communication times is mandatory for these systems, NoCs add another level of complexity. For soft RT systems, the number of missed deadlines can be substantially reduced by a routing mechanism where messages with tighter deadlines have higher priority. A core placement strategy based on message priorities and bandwidth requirements can also reduce the number of missed deadlines. The paper also discusses the impact of these strategies on energy consumption. We show that an interesting design space exists, where a right combination of strategies may result in an adequate trade-off between soft RT requirements and energy consumption.

We discusse related work in Section 2. Section 3 presents concepts about NoCs. Requirements of RT applications and their impact on NoC design are considered in Section 4. Section 5 introduces the approach proposed in this paper. The placement methodology is detailed in Section 6. Experiments and results are shown in Section 7. Section 8 gives concluding remarks and discusses future work.

2. RELATED WORK

High performance is obtained thanks to the available parallelism in the network, and scalability is achieved thanks to the network characteristic of regularity. Most of the works focus on the concept of using the regular NoC tile-based architecture, but they do not address the mapping problem.

Hu and Marculescu[6] present an algorithm to solve the mapping problem, by considering the communication energy consumption. This algorithm minimizes the energy consumption under specified performance constraints, but does not consider RT restrictions.

Bolotin[7] proposes a QoS NoC, where inter-module communication traffic is classified into four classes of service. Once communication requirements

of the target SoC have been identified, the network is customized. First, the modules are placed so as to minimize spatial traffic density. After, unnecessary mesh links and switching nodes are removed, and bandwidth is allocated to the remaining links and switches according to their relative load, so that link utilization is balanced.

In our work, we embed the QoS aspect into the mapping problem, in order to achieve QoS while looking at smaller power consumption.

3. NETWORKS-ON-CHIP

Several NoCs are described in the literature[2,3,4,8]. A NoC is composed by a set of routers and point-to-point links interconnecting routers in a structured way. Each router has a set of ports that are used to connect routers with their neighbor's routers and with the processing cores of the system.

NoCs are based on the message passing communication model. Cores in the network communicate by sending and receiving messages with a header, a payload, and a trailer. A NoC is described by its topology and by strategies for routing, flow control, switching, arbitration, and buffering. The topology is the arrangement of nodes and channels in a graph. Routing determines how a message chooses a path in this graph, while flow control deals with the allocation of channels and buffers to a message as it traverses this path. Switching is the mechanism that removes data from an input and places it on an output channel, while arbitration is responsible for scheduling the use of channels and buffers by the messages. Buffering defines the approach used to store messages when the router arbitration circuits cannot schedule them.

3.1 THE SoCIN NoC MODEL

SoCIN[9] is a scalable network based on a parametric router architecture to be used in the synthesis of customized low-cost NoCs. It uses wormhole packet switching and a deterministic XY-routing in order to ensure deadlock freedom at a low cost. The communication model uses message passing, and cores communicate by sending and receiving requests and responses. SoCIN uses wormhole packet switching[10]. A distributed approach for arbitration is used, and there is a round-robin arbiter at each output channel of a router. Flow control is based on a handshake strategy. Buffering is performed at the input channels, and there exists a p-words FIFO buffer at each input channel, where **p** depends on the application costs and performance requirements.

4. NOCS AND REAL-TIME REQUIREMENTS

Predictability represents the most important requirement for embedded applications with real-time restrictions. But other aspects of QoS, like performance, can be important too. Moreover, for embedded systems, application requirements (memory footprint, power) are more restrictive.

Embedded systems using NoCs may achieve reusability, scalability, and parallelism, while coping with power constraints. Furthermore, application requirements, such as timing restrictions and performance, can be met with the correct customization of NoC features. But the use of NoCs in RT systems adds complexity to the design, since the predictability of execution and communication time is mandatory. For predictability, it is necessary to know the time latency in the routers, since the communication time between two cores in the NoC must be smaller than the end-to-end deadline of the messages exchanged by them (i.e. from source to destination core).

In soft RT systems, a small number of deadlines may be missed, thus smoothly and temporarily degrading system performance. As shown in this paper, the number of missed deadlines in a NoC can be substantially reduced by the utilization of a routing mechanism based on message priorities, where higher priorities are given to messages with tighter deadlines. For hard RT systems, where deadlines cannot be missed, this mechanism must be eventually combined with a larger channel bandwidth. But it is important to notice that the required increase in bandwidth would be substantially larger in a hard RT system if there was no priority-based routing mechanism. Another important design aspect is the average time of the communications, which is directly related to the total energy consumption (of course the energy consumed in the core computations must also be considered). This average time may be reduced by a core placement strategy that maintains close to each other those cores that require a high communication bandwidth between them. Sometimes, however, these two requirements – reducing both missed deadlines and energy consumption – are conflicting, and an adequate trade-off may be found according to application requirements.

5. THE PROPOSED APPROACH

The proposed approach considers two relevant aspects: the core placement in the NoC and the flow control of messages by the arbiters of the routers. The main idea is to improve QoS of soft RT systems by reducing the number of missed deadlines, while simultaneously trying to reduce the average transmission time of messages and thus the energy consumption.

For the placement, we compare a random strategy with two other ones where communication requirements are considered. In the first one, a

straightforward approach assumes that the cost of a core in a given position is related only to the communication requirements with other cores. Cores communicating with a high rate should be placed nearby. In the second strategy, both the bandwidth and the priority (deadline-based) of the messages are considered. These strategies tend to decrease the average message transmission time, and thus the overall energy consumption.

Considering flow control, two policies are proposed besides the original round-robin arbitration mechanism of the SoCIN architecture. The first one is a priority-based arbiter, which is also based on message deadlines. The other alternative, besides priority-based flow control, uses multiple buffers at the output channels of the routers, allowing preemption of lower-priority messages by higher-priority ones. These priority-based mechanisms decrease the number of missed deadlines and thus increase QoS for soft RT systems. However, they also tend to increase the average message transmission time, since messages with lower priority may be blocked for larger time intervals.

Together, the arbitration mechanisms and the placement strategies build a design space that can be explored during the design of a NoC. For each application, with particular requirements considering task deadlines and energy consumption, an ideal combination of mechanisms may be found.

6. PLACEMENT ALGORITHM

Given a distributed application, it is necessary to determine the best placement of the processing cores over the network in order to match the application RT constraints, while reducing the total energy consumption.

To model the placement problem, we define an Application Communication Characterization (ACC) graph as the way the task communications occur. It can be described by the relationships among the application tasks, by the parallelism among message exchanges, and by the bandwidth and priority required by each message. This behavior can be modeled as an oriented graph, where vertices represent the application tasks, that send and receive message, and arcs express the relationship among tasks. The oriented arcs thus implement the paths for the messages and express the communication pattern of the application. The arcs are weighted in two alternatives. In the first one, the weight corresponds only to the bandwidth required by the message modeled by the arc. In the second one, the weight is given by the product between message priority and bandwidth.

Definition 6.1. ACC G=(V,A) is a directed graph, where each vertex $v_i \in V$ (i=1,2,...,m) represents an application task which sends and/or receives messages, and an arc $a_j \in A$ (j=1,2,...,n) is the directed path between the sender and receiver vertices of a message. For each directed arc $a_j=(u,v)$, a weight defines either the bandwidth $b(a_j)$ the application requires for the

message represented by a_j, or the product bandwidth*priority. A position $p(v_i)$ is assigned to each i vertex.

The communication architectures can also be modeled as a graph, whose vertices represent the routers in the NoC and the set of oriented arcs express all the communication channels given by the topology. This data structure is defined as the Architecture Communication Graph (ACG).

Definition 6.2. ACG is a directed graph, G'=(V',A'), where each vertex $v_q' \in V'$ (q=1,2,...,m) represents a router in a NoC and each arc $a_r' \in A'$ (r=1,2,...,1) is a channel between two routers that are directly connected. Furthermore, each router has a single local port, where a processing core is attached to. For each directed arc $a_r'=(u',v')$, a weight expresses the available bandwidth $b'(a_r')$ of the communication channel it represents. This parameter is taken from the network physical features such as channel width and frequency. The way the arcs are connected represents the topology.

In order to find a placement, one must map each application task (vertex of ACC) to a local port associated to a router (vertex of ACG).

Definition 6.3. Given an ACC and an ACG, for each vertex $v_i \in G$ in ACC there exists a corresponding vertex $v_q' \in G'$ in ACG, and vice-versa, i.e. there is a bijective mapping function $F: V \rightarrow V'$ s.t. $\forall v_i \in V, \exists v_q' \in V' (v_q'=F(v_i) \Lambda v_i=F^{-1}(v_q') \Lambda p(v_i)=p(v_q'))$.

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

Definition 6.4. A path $C = (v_1', a_1', v_2', a_2', ..., v_{m-1}', a_{m-1}', v_m')$ in ACG is an alternating sequence of vertices and arcs from the sender to the receiver of a message. A path is formed according to the routing strategy implemented in the network routers the ACG represent.

Using the above graph representations, the problem of matching the application real-time constraints, while reducing the total energy consumption under performance constraints, can be formulated as the problem of covering the rows of a m-row, n column, zero-one matrix $M = (a_{ij})$ by a subset j; j=1,...,n; of the columns $M_j=(a_{ij})$; i =1,...,m; at minimal cost. If we define $x_j=1$, if column j (with cost $c_j>0$) is in the solution, and $x_j=0$ otherwise, then the problem can be formulated as:

$$\text{Minimize} \quad Z = \sum_{j=1}^{n} c_j x_j \quad \text{subject to} \quad \sum a_{ij} x_j = 1 \qquad i = 1,...,m \tag{1}$$

$$x_j \in \{0,1\} \qquad j = 1,...,n \tag{2}$$

Equation (1) ensures that each row is covered by one column, and statement (2) is the integratility constraint. We can now formulate our placement problem as a set covering problem: M is the set of vertices in the

ACG used for each application message, $G'=(V',A')$; $M \subseteq V'$. This is so because all n messages of an application cover the set M of routers in a NoC, when they are sent over a network through their paths C.

Definition 6.5. A set $M_j=(a_{ij}=1 \mid i \in M)$ is a path C_j for one of the n messages in an application: $M_j = (C_j \mid v_q' \in C_j.\ a_{ij}=1 \wedge i=p(v_q'))$.

In our problem, c_j expresses the bandwidth (or the product between bandwidth and priority) the message M_j can reach. The semantic we adopted for c_j in the objective function says that if $c_j=0$, its associated bandwidth is equal to or greater than the bandwidth required by the application; otherwise, if $c_j=1$, the required bandwidth was not reached by the current placement.

Definition 6.6. Let $b'(a_r')$ be the bandwidth for a given arc a_r' in a path C_j, and B the required bandwidth for the message j:

$$c_j = 0, \text{ if } \min_{w=1}^{m-1} b'(a_w') \leq B; \qquad c_j = 1, \text{ otherwise}$$

As a consequence, minimizing the objective function means setting its value as closer to zero as possible. When Z is zero, all bandwidths were reached. This mapping problem has been proven to be NP-complete[11], and a number of heuristic solution algorithms have been presented in the literature. We adopted for this purpose the Simulated Annealing (SA) algorithm[12].

SA is a generalization of a Monte Carlo method for examining the equations of state and frozen states of n-body systems. The concept is based on the way liquids freeze or metals recrystalize in the process of annealing. In this process, a melt, initially at high temperature (T) and disordered, is slowly cooled so that the system at any time is approximately in thermodynamic equilibrium. As cooling proceeds, the system becomes more ordered and approaches a "frozen" ground state (T=0). SA has been used in various combinatorial optimization problems and has been particularly successful in circuit design problems[13].

7. EXPERIMENTS AND RESULTS

Fig. 1 shows our experimental setting. The application cores are placed in the NoC based on the traffic density of the communication (bandwidth) and on the time priorities of the application (deadlines). Synthetic task graphs are generated using TGFF[14] and used as input to the placement tool[15]. The placed NoC is then evaluated by a NoC simulation tool (NoCSim) in the timing aspects. NoCSim simulates the exchange of messages according to a previously defined communication bandwidth between two tasks. NoCSim returns performance parameters as the maximum and average times required to send each type of message. The deadline, as well as the bandwidth, is also defined for each communication, therefore NoCSim also provides the

number of messages that arrived after the deadline. NoCSim is specific to SoCIN, with the addition of multiple buffers in the routers to allow message preemption by priority, and has been developed in C++.

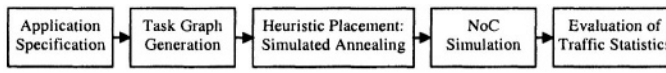

Figure 1. Experimental setting

To test impact of the placement and flow control strategies on the communication traffic, we carried out two experiments, using two sets of task graphs. The first set has tighter time restrictions than the second one, so that its deadlines are harder to respect. The design space is shown in Table 1.

Table 1. Strategies for placement and flow control

Placement:	**0** – without using SA (simulated annealing)
	I – with SA using only bandwidth as a weight
	II – with SA using bandwidth times priority as a weight
Flow control:	**R** – with round-robin arbiter
	P – with priority-based arbiter
	PP – with preemption (multiple buffers)

Results of the experiments, in terms of average message transmission time (which is directly related to overall energy consumption, measured in cycles) and percentage of missed deadlines, are shown in Table 2.

Table 2. Experimental results

		0-R	0-P	0-PP	I-R	I-P	I-PP	II-P	II-PP
	AvgTime	31.12	21.08	26.9	14.79	25.36	28.26	28.49	27.73
Experim.1	% deadline	35	23	25	33	18	15	17	17
Experim.2	% deadline	19	12	11	2	0.9	0.8	0.6	0.6

Following observations can be extracted from Table 2:

- If the bandwidth-based placement combined with round-robin is used (I-R), there is an average time reduction of 52.5% (from 31.12 down to 14.79) as compared to the original SoCIN network with arbitrary core placement (0-R). However, a reduction is not observed when a bandwidth-based placement is applied to NoCs using other arbiters.
- In the case of placement strategies applied with a priority-based arbiter (0-P, I-P, and II-P), there is a progressive reduction in the percentage of missed deadlines – from 23 to 18 and 17 in experiment 1, and from 12 to 0.6 in experiment 2. However, the average message transmission time increases (21.08, 25.36, and 28.49, respectively). The same behavior is noticed with a priority-based preemptive arbiter (0-PP, I-PP and II-PP).

As for the flow control, the use of a priority-based arbiter reduces the number of missed deadlines in comparison to the round-robin arbiter, both for random and bandwidth-based placements:

- From 0-R to 0-P there is a reduction from 35 to 23% of missed deadlines in experiment 1 and from 19 to 12% of missed deadlines in experiment 2;
- From I-R to I-P there is a reduction from 33 to 18% of missed deadlines in experiment 1 and from 2 to 0.9% of missed deadlines in experiment 2.
- However, the average transmission time increases from I-R to I-P by 71.5% (from 14.79 to 25.36).

When preemption is used in a priority-based arbiter, the number of missed deadlines decreases even more, but the average time increases again. For example, in the bandwidth-based placement the reduction (from I-P to I-PP) in the percentage of missed deadlines was aproximately 16% (18 to 15) and 11% (0.9 to 0.8) for the first and second experiments, respectively, but the average transmission time increased by 11,4% (25.36 to 28.26). By comparing II-P to II-PP, we see that a more complex arbiter with premption does not reduce the number of missed deadlines, since the priority-based placement already reduced this value to an apparent lower bound.

It can be concluded that the flow control with priority reduces the number of missed deadlines, but in general increases the energy consumption. The application of priority with preemption reduces even more the missed deadlines, but with an even higher energy consumption. In order to obtain a smaller increase in energy consumption in these cases, a new placement algorithm is required, considering the NoC dynamic behavior. The average transmission time increases because of messages that are blocked by other higher-priority ones: the placement algorithm should avoid communication bottlenecks in channels that are used frequently by high-priority messages.

It can also be observed that a prority-based placement has the same effect of reducing the percentage of missed deadlines as a priority-based arbiter, so that a more complex arbiter with premption is not useful in this case.

8. FINAL REMARKS

In this paper we discussed two possible alternatives to adapt NoCs for RT applications, where the correct predictability of execution and communication time is required. The first was the impact of the placement of the cores in the network in the behaviour of messages with priority. The other alternative considered the way how the flow control is made by the arbiter from the routers. We discussed a priority mechanism where the router would dispatch first the message with highest priority, or still if there could be preemption, when the router would have multiple queues of priorities.

We proposed a core placement strategy based on message bandwidth requirements and also on message priorities, in order to reduce the number of missed deadlines. The paper also had discussed the impact of these strategies on the energy consumption of the system. We had shown that an interesting design space can be explored, where the right combination of strategies might result in an adequate trade-off between soft RT requirements and energy consumption.

As future work we plan to use an adaptive routing to reduce the average message transmission time and therefore the energy consumption, while sustaining RT behavior. Another approach is adapting the placement mechanism for reducing the average time of message transmission when the priority-based arbiters are used.

REFERENCES

1. L. Benini, and G. De Micheli. Networks on Chips: a New SOC Paradigm. IEEE Computer, Jan. 2002, pp. 70-78.
2. P. Guerrier and A. Greiner. A Generic Architecture for on-Chip Packet-Switched Interconnections. DATE'2000, IEEE Press, 2000. pp. 250-256.
3. S. Kumar et al. A Network on Chip Architecture and Design Methodology. IEEE Computer Society Annual Symposium on VLSI, April 2002. pp. 105-112.
4. W. J. Dally and B. Towles. Route Packets, Not Wires: On-Chip Interconnection Networks. DAC'2001, ACM Press, 2001. pp. 684-689.
5. J. Duato et al. Interconnection Networks: an Engineering Approach. IEEE CS Press, 1997.
6. J. Hu and R. Marculescu. Energy-Aware Mapping for Tile-based NoC Architectures Under Performance Constraints. VLSI-SoC'2003, IEEE Press, 2003.
7. E. Bolotin et al. QNoC: QoS architecture and design process for Network on Chip. Special issue on Networks on Chip, The Journal of Systems Architecture, Dec. 2003.
8. A. N. F. Karim et al. An Interconnect Architecture for Networking Systems on Chips. IEEE Micro, Sep.-Oct. 2002, pp.36-45.
9. C. A. ZEFERINO and A. A. SUSIN, "SoCIN: a Parametric and Scalable Network-on-Chip". In: Proceedings of the 16th Symposium on Integrated Circuits and Systems (SBCCI'2003), São Paulo, Brazil, Sept. 2003, IEEE CS Press, pp.169-174.
10. W. J. Dally and C. L. Seitz. The Torus Routing Chip. Journal of Distributed Computing, Oct. 1986. pp. 187-196.
11. M.R. Garey and D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W.H. Freeman and Co., San Francisco, 1979.
12. N. Metropolis et al. Equation of State Calculations by Fast Computing Machines. J. Chem. Phys., 21, 6, 1953, pp. 1087-1092.
13. S. Kirkpatrick et al. Optimization by Simulated Annealing. Science, 220, 4598, 1983, pp. 671-680.
14. R. P. Dick, D. L. Rhodes and W. Wolf. TGFF: task graphs for free. Proc. Intl. Workshop on Hardware/Software Codesign, March 1998.
15. Blue Macaw, http://www.inf.ufrgs.br/~renato/bluemacaw, Apr 2004.