Regulation versus Flow Control in NoC for Hard Real-time Systems: a Preliminary Case Study

Hamdi Ayed, Jérôme Ermont, Jean-Luc Scharbarg, Christian Fraboul

Regulation versus Flow Control in NoC for Hard Real-time Systems: a Preliminary Case Study

Hamdi Ayed, Jérôme Ermont, Jean-luc Scharbarg, Christian Fraboul
Toulouse University - IRIT - ENSEEIHT
Toulouse, France
{hamdi.ayed2, jerome.ermont, Jean-Luc.Scharbarg, christian.fraboul}@enseeiht.fr

ABSTRACT

Many-core architectures are promising candidates for the design of hard real-time systems. Inter-core and core to external memory or peripheral communications use the Network-on-Chip (NoC). Such a NoC is typically composed of a set of routers. Internal organization of routers (mainly buffers) as well as flow control aspects impact NoC performances and thus those of the many-core, including the Worst-Case Traversal Time (WCTT) which has to be guaranteed for hard real-time systems.

In this paper we study the impact of flow control aspects on this WCTT. We consider two classes of NoC architectures, representative of the trend in the many-core market: Tilera Tile64-like NoCs where flow control is implemented at the router level and KalRay MPPA 256-like NoCs where flows are regulated at the source node level.

We compute flow WCTT for different configurations and we show that there is no clear winner, since NoC performances highly depend on flow features.

Keywords

Network-on-Chip, Flow control, Worst-case traversal time

1. INTRODUCTION

Many-core architectures are promising candidates to support the design of hard real-time systems. They are based on simple cores interconnected by a Network-on-Chip (NoC). Timing constraints, such as bounded delays, have to be guaranteed for hard real-time systems. Thus worst-case behavior of the NoC is a key feature for such systems.

However, the initial motivation when designing NoCs was to increase the average case throughput. NoCs can thus be used in hard real-time systems using one of the following approaches:

1. analysis of the Worst-Case Traversal Time (WCTT) of flows on existing many-cores,
2. modification of the hardware so that no contentions can occur by design, leading to straightforward WCTT for flows.

Several NoC have been proposed based on the second approach [3, 4, 9]. However, none of these NoCs targeting hard real-time constraints are available in commercially existing many-core architectures, such as for instance the Tilera Tile CPUs [10], the STMicroelectronics P2012/STHORM fabric [8] or the KalRay MPPA [1]. In this work, we focus on these commercially existing architectures, where NoC relies on wormhole switching [7] and Round-Robin Arbitration (RRA) within routers. Using wormhole switching, a packet is divided in flow control digits (flits) of fixed size which are transmitted one by one by routers. The header flit (i.e. the first flit) contains the routing information that defines the path for all the flits of the packet.

NoC implement flow control in order to control buffer occupancy. Two main strategies are considered in commercial many-cores. The first one implements flow control in each router: a packet cannot be forwarded if the next output port is busy. The Tilera Tile64 [10] uses this strategy. The second strategy implements flow regulation in source nodes, in order to bound the traffic. The KalRay MPPA 256 [1] uses this strategy.

The contribution of this paper is to evaluate the impact of these two strategies on flow WCTT. This preliminary evaluation is based on two small case studies.

The rest of the paper is organized as follows. Sections 2 summarizes considered NoC features. Section 3 presents the evaluation. Section 4 concludes and gives some direction for future work.

2. DESCRIPTION OF TWO NOC ARCHITECTURES

In this section, we describe two different NoC architectures: Tilera Tile64 and KalRay MPPA 256. The first one uses the classical credit based flow control. The second consists in a source regulation of flows.

2.1 Overview of the Tilera Tile64

The Tilera Tile64 is composed of a grid of 64 tiles. Each of them contains a processor engine (core), a private cache and a crossbar switch. The tiles communicate by exchanging packets through the embedded switch. To minimize the interference and to maximize the performance, inter-tile communication uses six independent networks. The traffic related to memory, caches, I/O and processors is transmitted upon distinct networks.

The Tilera Tile64 uses classical wormhole switching: packets are split into several flits and are transmitted flit by flit from the source to the destination tile. The first flit is called the header flit and contains the destination address. When the packet is granted to access to an output port, this output port is locked until the last flit has successfully traversed the switch. The flits follow the same path as the header flit. When the output port is locked, the flits are stored into a small sized (three flits) buffer of the input port, as shown in

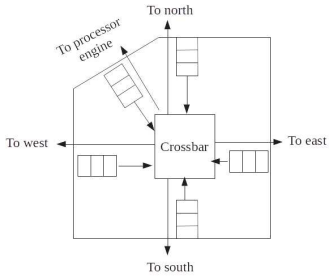

Figure 1: Tilera Tile64 router [10]

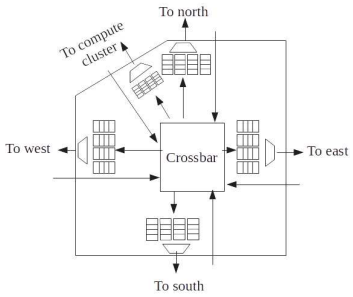

Figure 2: KalRay MPPA 256 router [1]

Figure 1. When the output port is freed, the switch fabric uses round-robin arbitration to ensure fairness. Thus, when the transmission of a packet from an input port is terminated, the transmission of a packet from another input port can start.

Control flow uses a credit scheme: the output ports contain a credit count corresponding to how many flits can be stored into the input ports of the next switch. Each time a flit is routed to the output port, the credit is decremented. When the credit count is zero, the flits are blocked into the input buffers. When an input buffer place becomes empty, the credit of the corresponding output buffer (from the previous switch) is incremented.

2.2 Overview of the KalRay MPPA 256

The architecture of the KalRay MPPA-256 is different from the one of the Tilera Tile64. It is composed of 16 processing elements (PE) which contain 16 cores each, and two parallel networks-on-chip, one for the data (D-NoC) and one for the control (C-NoC). The network topology is a 2D torus.

D-NoC is dedicated to high bandwidth data transfers. KalRay MPPA-256 uses flow regulation [6] at the source node. This regulation is parametrized by a window length (τ) and a bandwidth quota (β). At each cycle, the regulator compares the length of the packet to send plus the number of flits already sent during the previous τ cycles to β. If not greater, the packet can be sent, a flit each cycle. Using the network calculus theory, these parameters allow to determine the capacity constraints of the links and the router buffer sizes [1]. Consequently the NoC does not need control flow mechanism.

Table 1: Case study 1: flows set description

Flow	Period (cycles)	Length (flits)	T-bound (cycles)	K-bound (cycles)
f_1	1000	50	257	295
f_2	500	100	256	295
f_3	1000	50	278	304
f_4	1000	20	278	317

On NoC routers, flows can arrive from different directions. As shown in Figure 2, each direction has its own FIFO buffer at the output port. In that way, flows can be blocked only if they share the same output link. Round-robin is used to determine which packet in the FIFO queues is granted to be transmitted.

3. CASE STUDIES AND END-TO-END DELAY ANALYSIS

The goal of this section is to compare the WCTT on Tilera TILE64-like NoCs and KalRay MPPA 256-like NoCs. This comparison is based on two case studies.

3.1 Case study 1

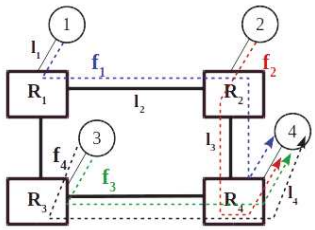

Figure 3: Case study 1: description

Figure 3 describes the first case study: a 2x2 Mesh NoC with 4 periodic flows (Table 1) transmitted to core 4. To compute WCTT for each flow, we use the Recursive Calculus [2] for Tilera NoC (T-bound) and Network Calculus [5] for KalRay NoC (K-bound). In the rest of this section, we illustrate these approaches for flow f_1.

3.1.1 Tilera NoC network

Several approaches have been proposed for WCTT computation for Tilera-like NoC architecture [2]. For this paper, we use the Recursive Calculus (RC), which gives tighter results than Network Calculus (NC) [2]. Figure 4 illustrates Recursive Calculus principle. For ease of presentation, it shows a simplified version of the worst-case scenario determined by RC approach for flow f_1. In this simplified version, the size of each input buffer is equal to one flit and packets size is 3 flits. This scenario can be easily generalized with packets of arbitrary size and larger buffers.

In this scenario, f_2 delays f_1 on link l_3. Thus, f_1 is blocked at R_2 till the end of transmission of f_2. f_2 is delayed on link l_4 by the packet with the maximum size coming from R_3 (l_3 in Figure 4). Due to round robin scheduling, f_1 is also delayed by one packet coming from R_3 (f_4 in Figure 4). This leads to the WCTT at the bottom of the the figure. Using

actual packet sizes of Table 1, we obtain $d^1_{e2e} = 7 * d_{sw} + 2 * L_3/C + L_2/C + L_1/C = 257\ cycles$.

It should be noted that f_3 is considered twice at l_4, since its packet size is larger than one f_4. Obviously, such a scenario is not feasible, due to f_3 period. Thus RC approach introduces some pessimism.

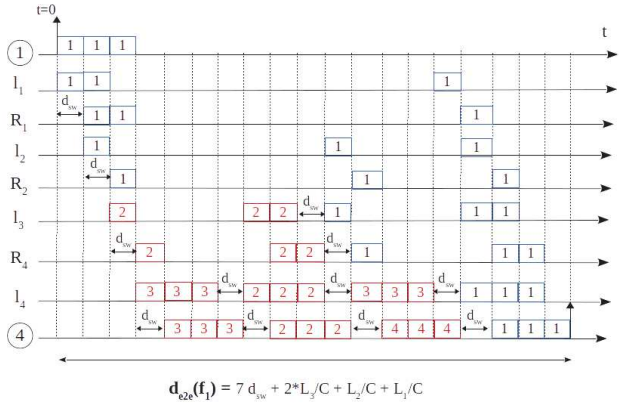

Figure 4: Case study 1: recursive calculus application for flow f_1 with Tilera NoC

3.1.2 KalRay NoC network

As presented in Section 2, flows are regulated at source node level and no flow control is applied at router level. Thus flits arriving in a router are stored in corresponding output port buffers (allocated to their input link). A packet-by-packet round robin scheduling is applied for each output port.

In this paper we consider a classical network calculus approach [5] for the WCTT analysis of flows. In such an approach, each flow f_i is modeled by an arrival curve α_i which overestimate its traffic. Each source node s_i or router output port R_{i_j} is modeled by a minimum service curve β_{s_i} or $\beta_{R_{i_j}}$. In the case study in Figure 3, each flow f_i is defined by a period T_i and a packet length L_i. Thus the arrival curve of a flow f_i is $\alpha_i(t) = \sigma_i + \rho_i * t$. σ_i is the maximum burst L_i and ρ_i is the maximum long term rate L_i/T_i. Arrival curves of the flows in Figure 3 are:

$$\begin{aligned} \alpha_1(t) &= \alpha_3(t) = 50 + 0.05 * t \\ \alpha_2(t) &= 100 + 0.2 * t \\ \alpha_4(t) &= 20 + 0.02 * t \end{aligned}$$

Concerning service curves, we assume that every link in the NoC has a transmission rate of $C = \frac{1\ flit}{cycle}$, the technical latency of source nodes is negligible and the technical latency of a router is equal to $d_{sw} = 1\ cycle$. Thus the overall service curves for a source node s_i and a router output port R_{i_j} are:

$$\begin{aligned} \beta_{s_i}(t) &= 1 * t \\ \beta_{R_{i_j}}(t) &= max(0, 1 * (t - d_{sw})) \quad i \in \{1, 2, 3, 4\} \end{aligned}$$

where d_{sw} represents the overall router latency.

Source nodes implement a First Come First Served policy while a round-robin scheduling is applied by router output ports. In order to deal with both policies we apply the (pessimistic) blind multiplexing model [5]: when two flows f_1 and f_2 share an output port R_{i_j} implementing any scheduling algorithm, the minimum service curve for flow f_1 is:

$$\beta^1_{R_{i_j}}(t) = (\beta_{R_{i_j}}(t) - \alpha_2(t))_\uparrow$$

where $\beta_{R_{i_j}}$ is the overall service curve offered by R_{i_j} and $\beta_\uparrow$ is the positive and non-decreasing upper closure defined as $(\beta)_\uparrow(t) = max(0, sup_{0 \le s \le t} \beta(s))$.

The WCTT computation for a flow f is based on the following result. When a flow f traverses two nodes R_{i_j} and R_{k_l} in sequence, offering service curves $\beta_{R_{i_j}}$ and $\beta_{R_{k_l}}$, respectively, network calculus theory [5] establishes that the concatenation of these two nodes offers the service curve $\beta_1 \otimes \beta_2$. $\beta_1 \otimes \beta_2(t) = inf_{0 \le s \le t} \beta_1(t - s) + \beta_2(s)$ is the Min-Plus convolution.

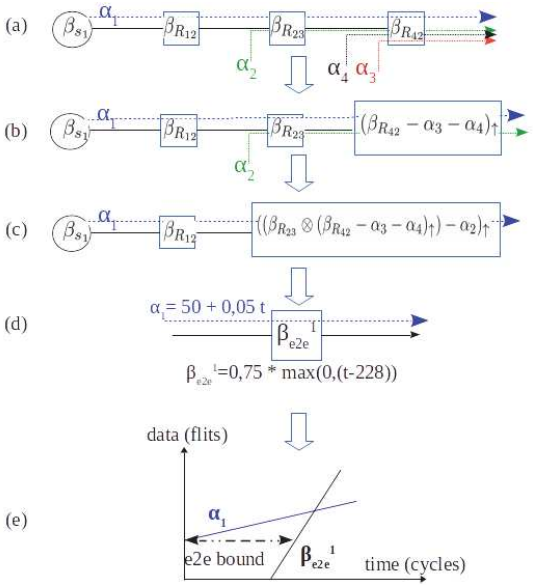

Figure 5: Case study 1: WCTT for flow f_1 with KalRay NoC

Figure 5 illustrate the WCTT computation for flow f_1:

- (a) first, service curves for source node s_1 and traversed output ports at crossed routers are computed as described previously in this section;
- (b) a service curve for aggregate flow $\{f_1, f_2\}$ at output port R_{43}: $(\beta_{R_{43}} - \alpha_3 - \alpha_4)_\uparrow = max(0, 1 * (t - d_{sw})) = max(0, 0.93 * (t - 77))$;
- (c) a service curve for flow f_1 offered by the system composed of the sequence $\{R_{23}, R_{42}\}$: $(\beta_{R_{12}} \otimes (\beta_{R_{43}} - \alpha_3 - \alpha_4)_\uparrow - \alpha_2)_\uparrow = max(0, 1 * (t - d_{sw})) = max(0, 0.75 * (t - 227))$;
- (d) an end-to-end individual service curve for flow f_1: $\beta^1_{e2e} = max(0, 0.75 * (t - 228))$;
- (e) as explained in [5], we can compute a bound on the WCTT of flow f_1 as the maximum horizontal deviation between α_1 and β^1_{e2e}: $h(\alpha_1, \beta^1_{e2e}) = 295\ cycles$

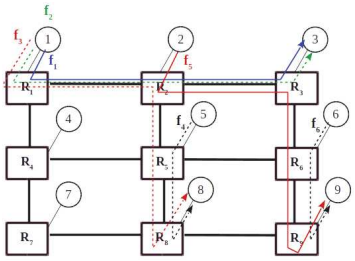

Figure 6: Case study 2: description

Table 2: Case study 2: flows set description

Flow	Period (cycles)	Length (flits)	T-bound (cycles)	K-bound (cycles)
f_1	1000	50	523	337
f_2	500	100	523	282
f_3	1000	50	774	276
f_4	500	100	154	190
f_5	1000	50	774	276
f_6	500	100	154	190

As we can see in Table 1, for case study 1 using Tilera Tile64-like NoC leads to lower WCTT bounds than KalRay MPPA 256-like NOC. All flows in this case study suffer only from direct blocking in routers, due to round robin arbitration. Using source regulation or flow control does not impact WCTT. The main reason for the differences in results of Table 1 is the worst-case delay computation, which is pessimistic for KalRay MPPA, because we overestimate round-robin impact. Removing this pessimism is an open problem, but it should lead to comparable results for KalRay MPPA and Tilera Tile. In such a situation Tilera Tile is probably the best choice, since the overall buffer size is smaller.

3.2 Case study 2

Figure 6 and Table 2 describe the second case study: a 3x3 mesh network with 6 periodic flows. Using the same approaches as described for case study 1, we computed WCTT for Tilera (T-bound) and KalRay NoCs (K-bound) and we reported them in Table 2.

As reported in Table 2, for case study 2, using KalRay-like NoC implies lower WCTT bounds for flows f_1, f_2, f_3 and f_5. The source node regulation strategy implemented by KalRay MPPA is clearly better for these flows. This is due to the fact that the considered configuration leads to indirect blockings when router flow control is used. These indirect blockings significantly increase flow WCTT. Conversely they do not impact WCTT when source node regulation is used. For flows f_4 and f_6, which do not suffer from indirect blocking, using Tilera Tile64-like NoC leads to lower WCTT bounds than KalRay MPPA 256-like NoC.

4. CONCLUSIONS AND FUTURE WORK

In this paper, we show that flow control implemented in NoC has a significant impact on flow WCTT. We compare source node regulation as implemented in KalRay MPPA and router flow control as implemented in Tilera Tile. We consider two small case studies. These two case studies show that flow WCTT highly depends on flow control strategy. We show that source node regulation leads to smaller WCTT for one case study while router flow control is better for the other one. It means that there is no clear winner and deeper studies are needed in order to be able to determine the most suitable flow control strategy from flow features. This should be based on a much larger evaluation of these strategies, based on representative case studies.

Up to now, we have considered very basic WCTT approaches. For instance, network calculus approach for KalRay MPPA doesn't make any assumption on flow scheduling, which might be very pessimistic. More elaborate approaches have to be considered.

Another question concerns the impact of buffer size on WCTT. To what extend can we decrease the flow WCTT by increasing buffer size?

5. REFERENCES

[1] B. D. de Dinechin, D. van Amstel, M. Poulhiès, and G. Lager. Time-critical computing on a single-chip massively parallel processor. In *Design, Automation & Test in Europe Conference & Exhibition, DATE 2014, Dresden, March 24-28, 2014*, pages 1–6, 2014.

[2] T. Ferrandiz, F. Frances, and C. Fraboul. A Sensitivity Analysis of Two Worst-Case Delay Computation Methods for SpaceWire Networks. In *Proc. of the 24th Euromicro Conf. on Real-Time Systems (ECRTS)*, pages 47–56, Pisa, Italy, July 2012.

[3] K. Goossens, J. Dielissen, and A. Radulescu. Æthereal network on chip: Concepts, architectures, and implementations. *IEEE Design & Test of Computers*, 22(5):414–421, 2005.

[4] A. Hansson, M. Subburaman, and K. Goossens. Aelite: A flit-synchronous network on chip with composable and predictable services. In *Proc. of the Conf. on Design, Automation and Test in Europe (DATE'09)*, pages 250–255, Nice, France, 2009.

[5] J. Leboudec and P. Thiran. *Network Calculus*. Springer Verlag LNCS volume 2050, 2001.

[6] Z. Lu, M. Millberg, A. Jantsch, A. Bruce, P. van der Wolf, and T. Henriksson. Flow regulation for on-chip communication. In *Design, Automation Test in Europe Conference Exhibition, 2009. DATE '09.*, pages 578–581, April 2009.

[7] L. Ni and P. McKinley. A survey of wormhole routing techniques in direct networks. *IEEE Transactions on Computers*, 26(2):62–76, Feb 1993.

[8] D. Rahmati, S. Murali, L. Benini, F. Angiolini, G. De Micheli, and H. Sarbazi-Azad. Computing accurate performance bounds for best effort networks-on-chip. *IEEE Transactions on Computers*, 62(3):452–467, March 2013.

[9] M. Schoeberl, F. Brandner, J. Sparsø, and E. Kasapaki. A statically scheduled time-division-multiplexed network-on-chip for real-time systems. In *Proc. of the Intl. Symp. on Networks-on-Chip (NOCS)*, pages 152–160, Copenhagen, Denmark, May 2012.

[10] D. Wentzlaff, P. Griffin, H. Hoffmann, L. Bao, B. Edwards, C. Ramey, M. Mattina, C.-C. Miao, J. F. B. III, and A. Agarwal. On-chip interconnection architecture of the tile processor. 2007.