PFT- A Low Overhead Predictability Enhancement Technique for Non-Preemptive NoCs

Bharath Sudev, Leandro Soares Indrusiak
Department of Computer Science
The University of York, U.K. YO10 5GH
Email: [bs638, leandro.indrusiak]@york.ac.uk

Abstract— **Performance predictability in Networks-on-Chip usually comes with high area and energy overheads. As an alternative approach, this paper presents a low overhead technique called Priority Forwarding and Tunnelling (PFT) which aims to enhance performance predictability in simple non-preemptive NoC routers. It addresses the negative impact of Head-of-line (HOL) blocking by temporarily boosting the priority of low priority packets that prevent the timely transmission of high priority packets. Further HOL blocking is prevented by opening priority tunnels across the NoC, preventing lower priority packets from acquiring output ports that will be needed by high priority packets in the near future. Extensive evaluation using the R3 NoC under different traffic loads shows the effectiveness of the proposed technique and quantifies the required hardware overheads.**

Keywords- Network-on-Chip, Predictability, Head-of-line blocking, Priority Forwarding and Tunnelling

I. INTRODUCTION

Over the years, advancement in chip technologies and the surge in processing element numbers elevated communication infrastructure as a major limiting factor as far as both performance and power consumption are concerned [1]. Though Network-On-Chip (NoC) is a prominent technology to address this issue, designing one has always been a trade-off between bandwidth, latency, power consumption and area overhead. In this paper, our focus is on the ability of a NoC to provide predictable communication performance, i.e. reduced variability in communication latencies.

To increase predictability, NoC designers have been using several techniques like preemptive arbitration [2], Virtual Channels (VCs) [3] and Time Division Multiplexing (TDM), all of which have high area and energy overheads. Though expensive, choosing not to use those techniques can have a negative impact on the NoC's predictability, as it makes it susceptible to Head-of-Line (HOL) blocking. As a result, high priority packet latencies can increase as they can fail to secure arbitration when low priority packets utilising the same communication channel are blocked down the line. This paper presents Priority Forwarding and Tunnelling (PFT), a low overhead predictability enhancement technique for non-preemptive NoCs, aiming to minimise the negative impact of HOL blocking and increase performance predictability. We demonstrate the effectiveness of the proposed technique within the R3 NoC, a simple priority-based non-preemptive architecture.

The paper continues with the review of some of the well-known NoC architectures and in Section III a typical HOL blocking scenario is depicted to motivate the improvements that are achieved through PFT. Section IV then details the proposed PFT technique and the R3-based evaluation setup is presented in Section V followed by simulation results in Section VI.

II. BACKGROUND

NoC is a promising communication infrastructure for multicores particularly due to the wide array of variable parameters presented to the designer, like topology, routing algorithm and switching strategy yielding a multitude of optimisation opportunities and its associated overheads. For example, the Hermes [4] NoC by Moraes et al was intended to be a low overhead and simple NoC architecture and hence it utilised wormhole switching and XY-routing making it extremely lightweight. Such a choice reduced the resource requirements to a bare minimum, but prevented the provision of packet latency guarantees, making it unpredictable.

QNoC, a predictability enhanced NoC was presented by Bolotin et al in [2] where packets can be provided with one of the four priority classes, and the priority value is used by the arbiter to deal with contention over output links. QNoC architecture featured preemption by which a higher priority packet requesting arbitration to a link is allowed transmission even if the link is engaged by a lower priority packet by preempting the lower priority transmission into buffers. QNoC does succeed in improving predictability of high priority packets but implementation of preemption increases significantly the area and energy overheads due to buffers and crossbars. Other approaches such as MANGO [5] use VCs to enhance predictability and improve latency figures under blocking. VC implementation costs both in logic hardware as well as buffering making the overall design bulky. This is quite clear in the work by Mello et al. in [6] where an advanced version of Hermes NoC with VCs was tested. It stated that while the Hermes with a single VC took 17% of the logic area of their hardware test-bed, the design with two and four VCs took 32.61% and 75.41% of the logic area respectively, which certainly cannot be neglected.

AEthereal [7] on the other hand implemented TDM as the means to bring about creditable predictability but as similar to VCs, TDM is expensive and the increase of NoC size results in the increase of TDM slot tables, thus limiting scalability. For enhancing predictability, approaches like [8], [9] and [10] used dynamically adaptive routing by monitoring traffic in the NoC in real-time. While Ge et al. in [8] utilised a centralised monitoring module to alter the source routing depending on the traffic on the NoC, Cidon et al. in [9] utilised traffic maps in their design for a similar mode of operation. Rantala et al. in [10] dealt with adaptability in a distributed perspective where the source routing at each network interface was altered depending on the congestion information retrieved from neighbouring routers. While these techniques deal with congestion in a defensive manner by avoiding routing of packets to congested routers, PFT provides more of an aggressive approach by confronting congestion head on.

III. Motivating Example

The evaluation platform for PFT is designated as the R3 NoC and it follows a five port router architecture based roughly around Hermes [8] hence employing XY-routing and wormhole switching to reduce hardware requirements. R3 NoC follows a mesh topology and unlike Hermes, each R3 packet header includes a priority value which is used by the arbitration unit of the router to resolve contention between packets over output ports.

One of the issues with such non-preemptive NoCs is that high priority packets could fail to secure arbitration if low priority packets already occupying the link are blocked down the line by other packets. This is called HOL blocking and a typical HOL blocking scenario is depicted in Figure 1 where boxes represent routers and arrows represent packets with the number inside the circles depicting its priority. Assuming 1 as the packet with the highest priority and that the priority decreases with increase in the numeric value (i.e. 2 less than 1, 3 less than 2 and so on), if all packets in the figure have destination south of the router (1,2), it can be observed that packets 3, 4 and 8 are withheld from securing arbitration as packet 9 is utilising the south port of router (1,2).

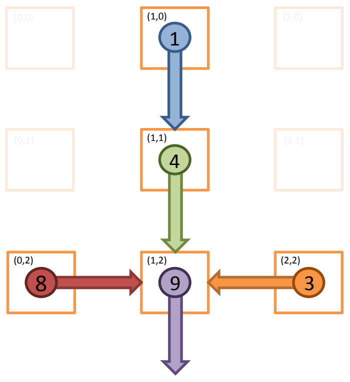

Figure 1 : Head-of-line blocking example

As packet 9 is of lower priority than others, it can be blocked down the line easily; hence indefinitely blocking higher priority packets like 1, 3, 4 and 8 up the line despite their higher priorities. Even if packet 9 goes through, the issue elevates further as packet 3 would get arbitration ahead of packet 4 forcing packet 1 to wait further up the line. When packet 3 finishes transmission, packet 4 would be transmitted followed by packet 8 ahead of 1 unless the router is designed to provide arbitration to packets immediately after the transfer of the preceding packet. As a result despite the highest priority value possible, packet 1 would have to wait until all the other packets get transmitted. Since all the other packets are susceptible to further blocking down the line due to their lower priority values, packet 1 is susceptible to have further waiting stages which could hamper its latency even more. Thus, under ordinary situation, the final transmission order of router (1,2)'s south port would be 9-3-4-8-1 (8 before 1 if arbitration in routers take more than a clock cycle) which goes against application-level priority assignment. Consider the case where the priority of packets sent from router (0,2) is 2 rather than 8. Under this situation, unless the period of packets from routers (0,2) and (2,2) is sufficiently high, packet 4 would never get arbitration as the packets from routers (0,2) and (2,2) would utilise the link in turns, one after the other hence blocking packet 1 up the line for ever.

R3 router utilises PFT to deal with all of the above issues hence ultimately enabling the south port of router (1,2) to transmit packets in the order 9-4-1-3-8. The use of PFT also prevents indefinite blocking of packet 9 and 4 down the line by other packets which would improve latency figure of packet 1 even further.

IV. Priority Forwarding and Tunnelling (PFT)

To resolve HOL-blocking, the first step as per PFT is to forward the priority of the high priority packet (blocked up the line) through the network until the blocked header of the low priority packet is encountered. Once the header is found, its priority is boosted to the priority of the high priority packet so that the block is resolved. For enhanced performance, the output port in the path that will be used by the high priority packet in the future will be tunnelled (locked) for that specific priority value such that packets with lesser priority will not be granted arbitration to that port temporarily until the high priority packet is transferred.

As an example, the functionality of a PFT-enabled R3 NoC is shown in Figure 2, detailing the internal design of routers (1,1) and (1,2) under the scenario depicted in Figure 1. Whenever a R3 packet gets blocked and the flow that caused the block is blocked as well, its destination address and priority value are stored into a 'local blocking data' register called $\propto$-register as seen inside the north port of router (1,1) (where packet 1 is blocked). This information is sent through the network to the router to which the port is requesting arbitration towards and when the next router receives this data, it is stored into its 'remote blocking data' register called β-register (as shown inside the north port of router (1,2)). Similarly, this information is transferred from router to router until it reaches the header of the blocked packet (down the line) and if the arbitration request priority of the blocked packet is less than that of the blocking information carried in the corresponding β-register, the priority of the request is boosted to that of the packet blocked up the line (in this case, priority of arbitration request of packet 4 boosted to 1 at north port of router (1,2)).

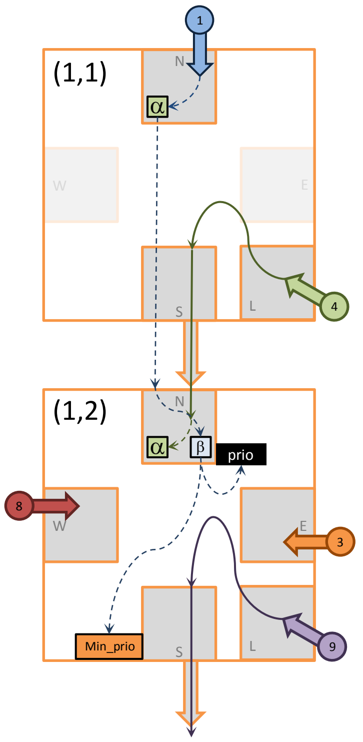

Figure 2 : PFT operation example

In this example, this allows packet 4 to obtain arbitration to the south port of router (1,2) ahead of packet 3 despite its lower priority value. To prevent other packets from securing arbitration to the south output port of router (1,2), R3 also preforms what we call tunnelling by which the minimum priority for arbitration to a specific output port can be set. In the example, the south port is tunnelled to priority 1 which would prevent packet 8 from securing arbitration before packet 1 if the design takes more than once clock cycle for arbitration. This lockable arbitration feature is built into the arbitration unit which enables R3 routers to prevent further HOL blocking on those specific routers by other packets.

This tunnelling of output port may or may not be done to the same port to which the packet blocked down the line (packet 4 in this case) is requesting arbitration to. In the current example, since packet 4 and 1 have the same destination, the same path of packet 4 is tunnelled for packet 1. Consider the case if the destination of packet 1 is (1,2). Then packet 1 would be utilising the local port of (1,2) after packet 4 is transmitted and hence the local port would be the one that has to be tunnelled in advance. This is why the destination information of the packet blocked up the line is send as part of the blocking data so that the R3 routers down the line can determine the appropriate router and its appropriate output port to tunnel.

In the actual implementation, each input port of each router would have a ∝-register and each input port apart from the local port would have a β-register though it's not shown in Figure 2 for simplicity. Similarly, to lock output ports for specific priority values, each output port would have a register to hold such information if necessary.

α-registers and β-registers would be serviced one at a time in round robin fashion such that if the required operation is not possible (due to busy connection lines or registers), that register is skipped and the next register is serviced. For lower hardware overhead, there is another low performance version of R3 featuring TDM for servicing the registers. As transactions do not occur when packets are blocked, the data links itself could be used to transfer α-register and β-register data in-between routers hence reducing any additional overhead of extra connection lines. Although the current R3 implementation (R3B4) uses dedicated lines, future versions are planned without those for lower hardware overhead.

V. Evaluation Setup

The evaluation prototype based on the R3 NoC was designed in Bluespec and simulated using BlueSim simulator. The setup consists of the router design enveloped in a parameterisable test-bench which replicates and interconnects the routers according to a 2D-mesh topology. The local port of each router was connected to packet-generators which could be pre-set with parameters like start time, period, packet size, priority and destination. Apart from conditioning the data to the required flit format and injecting into the NoC, the packet-generators also receive packets from the NoC and export evaluation figures to an external file. For testing the performance of PFT under different load conditions, we use average load per link (V) as the metric to classify the different evaluation scenarios used. For a fixed task mapping, as the total load in the NoC could be found by summing the ratio of total transmission time and period of each kind of packet, dividing it by the number of links would provide the average load per link (V) as shown in Equation 1.

$$V = \left\{ \sum_{x=0 \; to \; W-1} \sum^{y=0 \; to \; H-1} \left(\frac{C_{x,y} \times H_{x,y}}{P_{x,y}} \right) \right\} \times L_{(W \times H)} \qquad \text{Eq. 1}$$

(W- NoC Width, H- NoC Height, C- Latency, H- Hops, P- Period, L- Number of links)

As per the equation, V increases with the load in the NoC and a 4x4 NoC was tested with and without PFT under different values of V by adjusting the periods of packets.

VI. Implementation Result

The latency figures of the NoC with and without PFT under V=0.464 and V=0.755 are presented as box-plots in Figures 3 and 4 respectively. The box-plot whiskers (thin lines connected to each box) represent the extreme cases of latency (hence best and worst case) and the boxes enclose the upper and lower quartile of latency with the middle line depicting the median. For a specific packet, the closer the boxplot is to the X-axis the lower the latency and the shorter the box of the boxplot, the lower the variability in latency.

As contentions would have been rare events, at extremely low load conditions, PFT provided only minor improvement in latency and variability for V=0.464 (Figure 3). When the NoC was loaded with higher intensity traffic (V=0.755) contention situations increased and hence the performance as seen in Figure 4.

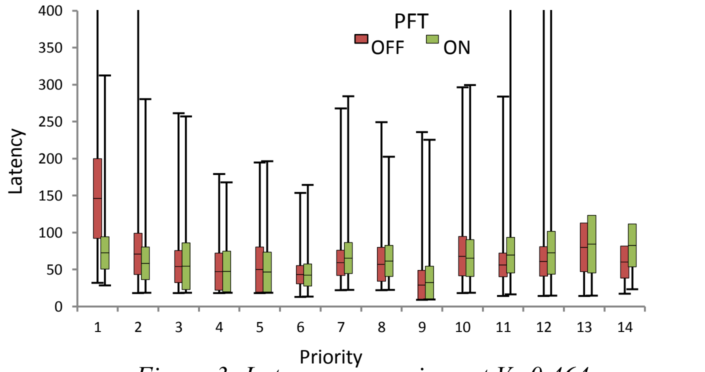

Figure 3: Latency comparison at V=0.464

In Figure 4, it can be seen that when PFT was switched on, for high priority packets (packets 1,2,3,4 and 5), the box and whiskers got lower and shorter depicting lower latency and lesser variability at the cost of low priority packets (like packets 10 and 11).

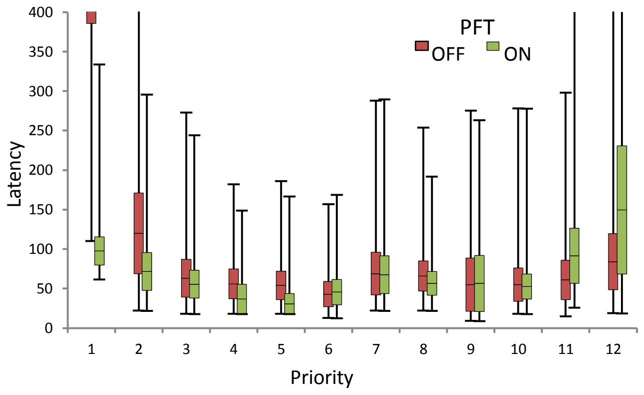

Figure 4: Latency comparison at V=0.755

When the NoC was put under heavy traffic condition (V=1.54), the ill effects of HOL blocking becomes critically prominent. Without engaging PFT, high priority packets 1 and 2 get blocked indefinitely and as a result, less than two packets of priority 1 and 2 got through the network in 10^5 clock cycles while hundreds of packets with lower priority reached their destination. This can be seen in Figures 5 where the cumulative frequency of packet reception numbers is plotted. In the plot, it can be seen that the line depicting the performance with 'PFT ON' is ahead of the other line for high priority packets (packets 1 to 10) hence depicting higher reception numbers.

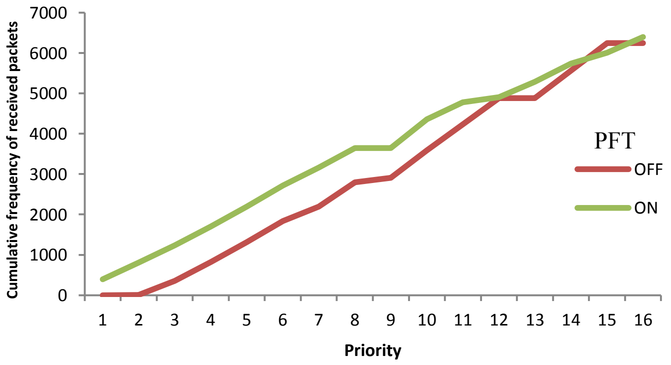

Figure 5: Packet reception Cumulative Frequency at V=1.54

Despite the performance figures, PFT logic has modest hardware requirements. On evaluation with Xilinx tools, it was verified that the overhead of R3 router with PFT logic was 8% adder/subtractors, 27% registers and 142% comparators compared to the R3 router without PFT logic.

VII. Conclusion

To increase predictability of non-preemptive NoCs, this paper introduced the PFT technique which counters the ill effects of HOL blocking. The paper also presented the R3 NoC, the evaluation prototype for PFT and its implementation and operational details along with preliminary test results. Under normal load conditions, PFT provided lower latency and variability than the baseline NoC architecture and under heavy load conditions; it provided increased responsiveness by preventing high priority packets from getting blocked indefinitely by low priority packets. Though the performance improvement is significant as shown by the simulation results, the hardware overhead associated with PFT logic on the R3 NoC is modest. Future work will optimise the technique even further by allowing the PFT control messages to use the full bandwidth of the NoC links. This can be done without any impact on performance, as PFT is only used over paths that are completely blocked and thus not utilised.

VIII. Acknowledgement

This work is supported by EPSRC project LowPowNoC (EP/J003662/1) and EU FP7 project T-CREST (288008).

IX. References

[1] J. Henkel, W. Wolf, and S. Chakradhar, 'On-chip networks: a scalable, communication-centric embedded system design paradigm', *VLSI D. Proceedings*, 2004, pp. 845–851.

[2] E. Bolotin, I. Cidon, R. Ginosar, and A. Kolodny, 'QNoC: QoS architecture and design process for network on chip', *Journal of Systems Architecture*, vol. 50, no. 2–3, pp. 105–128, Feb. 2004.

[3] W. J. Dally, 'Virtual-channel flow control', in *Proceedings of the ISCA*, New York, NY, USA, 1990, pp. 60–68.

[4] F. Moraes, N. Calazans, A. Mello, L. Möller, and L. Ost, 'HERMES: an infrastructure for low area overhead packet-switching networks on chip', *Integr. the VLSI Journal*, vol. 38, no. 1, pp. 69–93, Oct. 2004.

[5] T. Bjerregaard and J. Sparso, 'Implementation of guaranteed services in the MANGO clockless network-on-chip', *Computers and Digital Techniques, IEE Proceedings* -, vol. 153, no. 4, pp. 217 – 229, Jul. 2006.

[6] A. Mello, L. Tedesco, N. Calazans, and F. Moraes, 'Virtual channels in networks on chip: implementation and evaluation on hermes NoC', in *Proceedings of the SBCCI*, NY, USA, 2005, pp. 178–183.

[7] K. Goossens, J. Dielissen, and A. Radulescu, 'AEthereal network on chip: concepts, architectures, and implementations', *IDTC, IEEE*, vol. 22, no. 5, pp. 414 – 421, Oct. 2005.

[8] F. Ge, N. Wu, and Y. Wan, 'A network monitor based dynamic routing scheme for Network on Chip', in *Microelectronics Electronics, 2009. PrimeAsia 2009. Asia Pacific Conference on Postgraduate Research in*, 2009, pp. 133 –136.

[9] R. Manevich, I. Cidon, A. Kolodny, I. Walter, and S. Wimer, 'A Cost Effective Centralized Adaptive Routing for Networks-on-Chip', in *DSD*, 2011 pp. 39 –46.

[10] V. Rantala, T. Lehtonen, P. Liljeberg, and J. Plosila, 'Distributed Traffic Monitoring Methods for Adaptive Network-on-Chip', in *NORCHIP, 2008.*, 2008, pp. 233 –236.

[11] F. Moraes, N. Calazans, A. Mello, L. Möller, and L. Ost, 'HERMES: an infrastructure for low area overhead packet-switching networks on chip', *Integr. VLSI J.*, vol. 38, no. 1, pp. 69–93, Oct. 2004.