Low Overhead Predictability Enhancement in Non-preemptive Network-On-Chip Routers using Priority Forwarded Packet Splitting

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

Abstract— **Providing predictability enhancement for NoC packets can be a difficult proposition when dealing with simple non-preemptive designs primarily due to occurrence of head of line blocking and tail backing of high priority packets as a result of lower priority packets. Typically, predictability is enhanced by making the NoC preemptive by the use of Virtual Channels or employing techniques like Time Division Multiplexing which are generally expensive. This paper presents a low overhead predictability enhancement approach for non-preemptive NoCs which utilises low overhead techniques to resolve head of line blocking and tail backing. As per the technique, head of line blocking is resolved by enabling the low priority packet causing the block to inherit the priority of the blocked high priority packet while tail backing is resolved by splitting the low priority packet that cause the tailback. We then demonstrate the effectiveness of the technique using a prototype and evaluate the implementation overhead.**

Keywords- Network-on-Chip, Predictability, Priority Forwarded Packet Splitting, Non-preemptive Network-on-Chip, PFS.

I. Introduction

Packet predictability in Network-On-Chip (NoC) designs is usually enhanced by utilising techniques like Virtual Channels (VC) [1] or Time Division Multiplexing (TDM) [2] both of which result in significant hardware complexity and buffer utilisation, thus resulting in increased area and power requirements. Considering predictability enhancement as the reduction of variability in latency [3], we present a low overhead technique to improve predictability of simple non-preemptive NoCs by resolving head of line (HOL) blocking and tail backs which affects non-preemptive NoC packets.

HOL blocking and hence the resultant tail back is a significant factor degrading non-preemptive packet predictability and quoting Huang et al from [4] “due to HOL blocking, the throughput of the links is typically limited to 58% under uniform traffic with fixed packet length”. The Priority Forwarded Packet Splitting (PFS) technique we present here, employs Priority forwarding [5] to resolve HOL blocking in routers so that when a high priority packet is blocked by a low priority packet, the low priority packet would be forwarded with the priority of the high priority packet thereby resolving the block. Tailbacks are resolved by employing a low performance low overhead emulation of preemption called Selective Packet Splitting (SPS) [6] using which lower priority packets causing tailbacks are split (rather than preempted) so as to allow the high priority communication to occur immediately. Both of these techniques are relatively inexpensive and complimentary thereby providing predictability enhancement for packets without major overheads. We demonstrate the effectiveness of PFS within a simple non-preemptive NoC architecture and evaluate the resultant implementation overhead.

The paper continues with the review of literature in Section II followed by Section III where a HOL blocking scenario is provided as an example to motivate the improvements that can be achieved with the technique. Section IV then details the PFS approach along with the prototype implementation details. The implementation results are added in Section V followed by conclusion, acknowledgment and references as subsequent sections.

II. Background

As NoCs provide a wide variety of tuneable parameters for the designers, there have been several architectures developed over the years each presenting a specific kind of trade-off between benefit and overhead. The simplest of them all is the Hermes [7] NoC by Moares et al which is a lightweight design utilising wormhole switching and XY routing with simple round robin arbitration. This simplicity came at the cost of prioritisation for packets and hence lacked provision for performance guarantees. On the contrary, packet predictability was ensured in AEthereal [8] NoC but employing TDM thereby making the NoC almost completely predictable. Though TDM ensured the timeliness of the packets in this case it made AEthereal bulky and as TDM utilises slot tables, it limited the scalability and link utilisation.

Another means of predictability enhancement frequently employed by NoC designers is the use of Virtual Channels like seen in QNoC [9] and MANGO [10]. With virtual channels, the routers would be able to transmit a higher priority packet even when the link is being used by a lower priority packet. This was enabled by utilising buffers to preempt the low priority flow so that the higher priority packet can be transmitted as soon as possible. VCs are also used in commercial NoCs like seen in Tilera's [11] processors and NetSpeed [12] designs but the use of VCs bring about a heavy overhead of buffers and associated hardware. This effect is visible in the work by Mello et al in [13] where a Hermes NoC with Virtual Channels was evaluated. The paper stated that while the Hermes NoC 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. Link Division Multiplexing (LDM) [14] is a similar technique where the physical link is multiplexed to transfer multiple packets simultaneously. Though LDM needs fewer buffers than TDM or VCs, it also results in significant hardware overhead. Most of the above techniques depend broadly on the use of buffers and this can be infeasible for lightweight embedded systems as buffers cost in area and power. For example; in [15] Kundu points out that buffers alone amount to 15% of the area of their basic router and account for 46% of the total power consumption. For embedded applications having resource restrictions, such excessive resource requirements can be quite infeasible.

Other predictability enhancement approaches use adaptive routing based on the traffic scenario in the NoC [16], [17], [18]. While Ge et al. in [16] utilised a centralised monitoring module to dynamically alter the source routing, Cidon et al. in [17] utilised traffic maps in their design for a similar mode of operation. Rantala et al. in [18] 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.

With this paper, we investigate the use of priority forwarding coupled with packet splitting in routers so that HOL blocking and tail backing can be resolved, thereby improving packet priority without using expensive priority enhancement techniques or adaptive approaches. Since the techniques utilised here are relatively simple and less hardware intensive; the resulting routers would be light weight and hence more suitable to be used in lightweight embedded systems.

III. Motivating Example

To demonstrate the issues that can affect simple priority based NoC packets, a traffic scenario is depicted in Figure 1 where boxes represent routers, arrows represent packet flows, while the number inside the circles depicts packet priority. Assuming 1 as the packet with the highest priority and that the priority decreases with increase in 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 2, 3 and 5 are withheld from securing arbitration as packet 7 is utilising the south port of router (1,2).

As packet 7 is of lower priority, it can be blocked down the line easily; hence has a potential to block the other packets (1, 2, 3 and 5) up the line despite the higher priority values. Even after packet 7 gets transmitted, the issue would prevail as packet 2 would get arbitration ahead of packet 5 forcing packet 1 to wait further up the line. Followed by the transmission of packet 2, packet 3 would be transmitted while packet 1 remains detained behind packet 5. As a result it would be after the transmission of packet 3 and 5 that packet 1 would be transmitted eventually.

So, despite the highest priority value possible, packet 1 would have to wait until all the other packets get transmitted thus increasing its latency. Since the packets that would secure arbitration before packet 1 would be susceptible to further blocking down the line due to their lower priority values, packet 1 is susceptible to further waiting stages which could worsen its latency even more thus rendering application level priority assignment pointless.

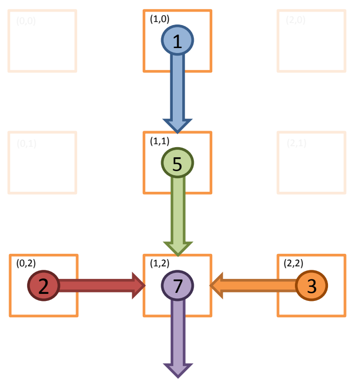


Figure 1: Blocking Example

In the above scenario, we can observe that packet 1 is disadvantaged due to primarily two reasons: HOL blocking and tail backing.

Assume that the packets have 100 flits each. As a result of HOL blocking; packets 2 and 3 would secure arbitration before packet 5 and hence the latency of packet 1 would go up by at least 200 clock cycles (size of packet 2 + size of packet 3) if the routers take one cycle per flit.

As packet 5 is ahead of packet 1 causing tailback and packet 7 is further down causing more tailback, a further delay of at least 200 clock cycles (size of packet 5 + size of packet 7) would be encountered bringing up the total delay to 400 clock cycles without accounting for arbitration delay and interference from any other traffic on the NoC.

IV. Priority Forwarded Packet Splitting

PFS logic utilises priority forwarding to deal with HOL blocking of packets and hence whenever a packet gets blocked by a blocked lower priority packet, the priority of the high priority packet (blocked up the line) would be forwarded down the line to the blocked low priority packet header. This would enable the low priority header to assume the priority of the high priority packet temporarily hence resolving the HOL blocking scenario.

In the example in Figure 1, as packet 1 gets blocked by the blocked packet 5, the priority of packet 1 would be assigned to the header of packet 5 by employing priory forwarding. This would enable packet 5 to secure arbitration ahead of packet 2 and 3, and hence eliminating the latency surge owing to those two packets.

Tailbacks are resolved by employing selective packet splitting technique so that higher priority packets behind the line would be able to secure arbitration by splitting the low priority packet (that is causing tailback) effectively pre-empting the low priority packet using minimal hardware overhead.

In the example, as packet 5 is arbitrated to the south port of router (1,1) and as the higher priority packet 1 is destined to use the same port, the packet splitting logic would be triggered thereby splitting packet 5 and hence terminating the already arbitrated connection with a tail flit. The rest of packet 5 would be then provided with a new header and a new arbitration request would be imitated at router (1,1). As priority forwarding would already have updated the priority of the header of packet 5 to 1 in router (1,2), router (1,2) would then be able to split the packet which would be utilising the south port then so that the split packet 5 (of the width of a few flits) can be transmitted immediately followed by the entire bulk of packet 1.This would enable packet 1 to split packet 5 and packet 7 into two separate packets, thereby allowing packet 1 to secure arbitration to the south port before the other packets get transmitted completely.

Assuming the input buffer depth to be 2, this would theoretically allow packet 1 to secure arbitration in under 20 clock cycles if further blocking doesn't occur down the line.

A. Prototype Implementation

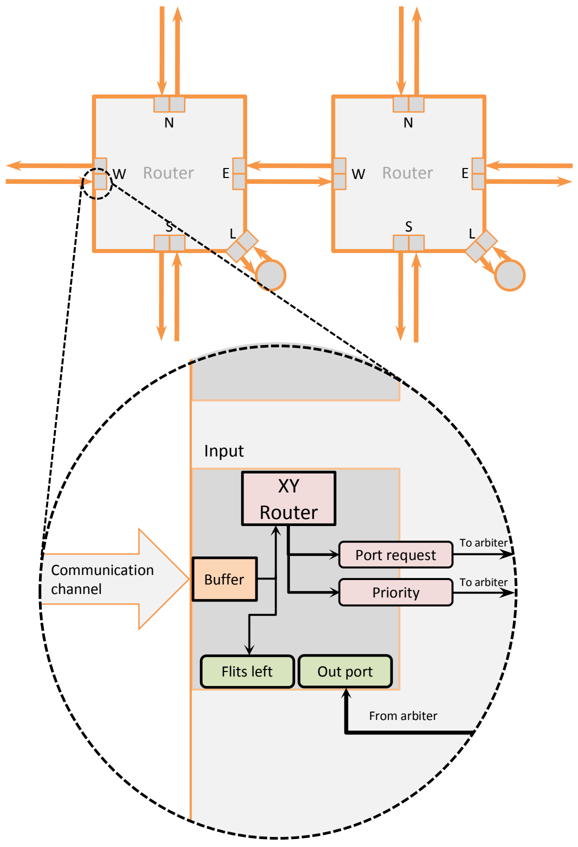


Figure 2: Basic Router Design

The prototype router was designed as a five port architecture based roughly around Hermes hence employing XY-routing and wormhole switching to reduce hardware requirements. The design uses a uniform mesh topology and unlike Hermes, each packet header includes a priority value which is used by the arbitration unit of the router to resolve contention between packets over output ports.

As shown in Figure 2, the routers have buffered input ports which on reception of a packet header employ XY-routing to set the 'port request' register and the 'priority' register in accordance with the destination and priority information carried. The arbitration unit in the router then checks 'port request' and 'priority' registers of all input ports to provide arbitration to the qualified ports.

The arbiter then establishes connection by setting the 'out port' and 'flits left' registers on the input port. This permits the input port to send flits to the allocated output port so that the flits could be transferred away through the communication links. As the flits are being transferred, the input port also decrements the value in the 'flits left' register so that when the value reaches zero, the connection can be closed by re-setting the 'out port' register value to zero.

1) HOL blocking resolution using Priority Forwarding

The aim of priority forwarding is to resolve HOL blocking by forwarding the priority of the high priority packet (up the line) to the blocked low priority packet down the line. So, when a packet gets blocked by a blocked packet of lower priority, the priority of the high priority packet would be loaded into a local blocking register called **α** register.

To load local blocking information, each of the five input ports are provided with α registers capable of storing a priority value each. The internal functionality of the routers in the

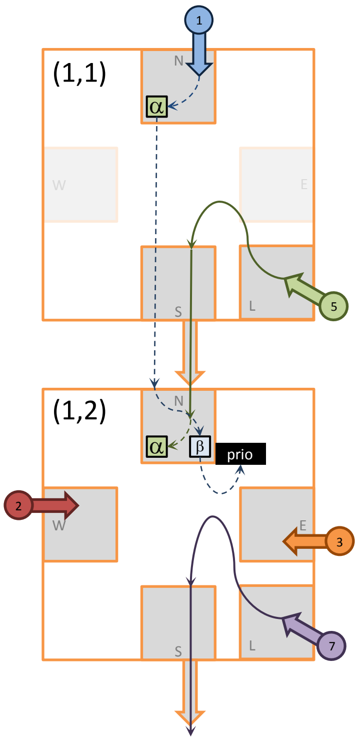


Figure 3: Priority Forwarding Logic Implementation

previous example is depicted in Figure 3 where it can be seen that as packet 1 is blocked by the blocked packet 5, the priority of packet 1 is stored into the α register inside the north port of router (1,1).

The information inside the α register would be then forwarded down the line through the blocked path to the next router and every time it reaches a new router, it would be stored into a remote blocking register called β register. Each input port other than the local port would have β registers to store such blocking information coming from nearby routers.

α and β registers would be serviced in a round robin fashion and every time a β register is serviced the router would check for the presence of the blocked header and if the header is further down the line, the information would be forwarded to the next router until the blocked header is reached.

Once the header is found and the corresponding β register is serviced, the arbitration request priority will be updated by altering the value inside 'priority' register inside the corresponding input port. As priority forwarding only modifies the arbitration request (instantaneously to resolve blocks) and not the packet itself, there would be no risk of low priority packets assuming high priorities permanently hence affecting higher priority packets negatively.

In the current example, priority value 1 would be forwarded from the α register in router (1,1)'s north port to the β register in the north port of router (1,2), hence ultimately enabling the router to modify the value in the 'priority' register, thus providing the arbitration request of packet 5 with the priority value 1.

Though the current prototype uses dedicated links to transfer the blocking information between routers, the same data links can theoretically be used for the purpose as the data links would be idle when they are blocked. This would require implementation of a credit based flow control system to prevent input buffers from getting blocked however further experiments has not been done in this respect and is considered as future work.

2) *Tailback resolution using Selective Packet Splitting*

Selective packet splitting aims to improve packet predictability by enabling routers that encounter preemptible scenarios to split the associated low priority packet so that the high priority packet can be transmitted as soon as possible. The decision to split a packet would depend on two parameters: priority difference (PD) between competing packets and the number of remaining flits (RF) from the low priority packet. Both parameters can be assigned statically or varied dynamically according to the required magnitude of slack in predictability. These two parameters would allow the NoC to vary its effort in predictability enhancement depending on the criticality of scenario.

If a packet satisfies these two conditions, the router would split the transmission by sending a tail flit followed by the construction of a new header which would then initiate a new arbitration request at that router. This would allow the higher priority packet (which would already have requested arbitration) to secure arbitration immediately and hence reduce latency.

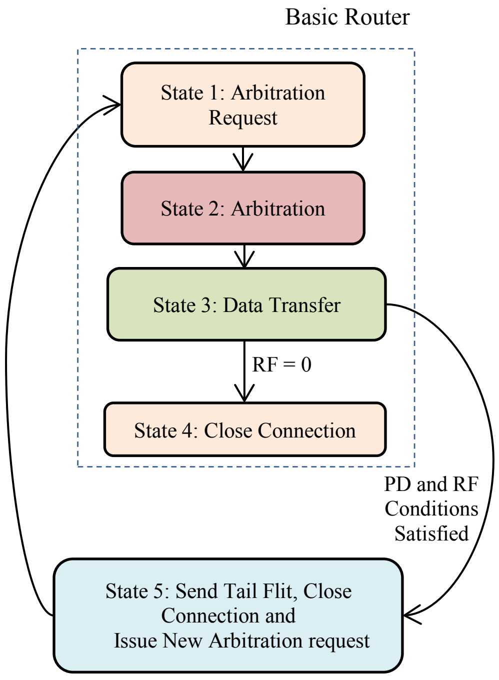

Basic Router
State 1: Arbitration Request
State 2: Arbitration
State 3: Data Transfer
RF = 0
State 4: Close Connection
PD and RF Conditions Satisfied
State 5: Send Tail Flit, Close Connection and Issue New Arbitration request

Figure 4: Splitting Operation

As shown in Figure 4, the state machine that manages the different stages of communication (arbitration request, arbitration, data transfer and close connection) has an extra state to perform packet splitting depending on the parameters set. So, once the conditions are satisfied for packet splitting, the input port concludes the active low priority communication with a tail flit and then releases the associated output port by setting the respective 'out port' register to zero. Simultaneously with the termination of the communication the input port issues a new arbitration request for the splitted packet so that once the high priority communication is completed, the low priority communication can be resumed. To enable this, each input port would have a register to store a newly constructed header and the most significant bit of every flit is used to denote a tail flit so that the routers would be able to identify termination of a packet.

V. Implementation and Results

The evaluation prototype was designed in Bluespec System Verilog and was developed as a router design enveloped in a parameterisable test bench which was used to replicate and interconnect the routers according to a 2D-mesh topology. The local port of each router was connected to packet-generators which could be set with packet parameters like start time, period, packet size, priority and destination according to which packets would be generated.

The packet-generator configuration is auto-generated as Bluespec source code using a custom built code generator which could either configure the generators randomly or in accordance with a series of algorithms to generate specific configuration patterns.

The packet generators also include logic to receive packets from the NoC and export evaluation figures to an external file. Every time a packet generation or reception occurs, the event is documented an entry onto the external data file so that our custom built macro code running inside the spread sheet software would be able to analyse it to generate latency statistics and graphs.

A. Performance

As PFS does not employ separate buffers to deal with each of the service levels, NoCs utilising PFS can be scaled without issues. We used average load per link (V) (detailed in Eq. 1) as the measure of the load on the NoC and a 4x4 NoC was analysed with different traffic scenarios to extract performance statistics. 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.

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

(W- NoC Width, H- NoC Height, D- No load latency, H- Hops, P- Period, L- Number of links)

The latency performance of the technique is interpreted in the paper as boxplots with priority of the packet on the X-axis and latency (in clock cycles) on the Y-axis. In box plots the whiskers shows the extreme cases of latency and the boxes indicate the upper and lower quartile of latency with the middle line depicting the median. So, shorter box and whiskers show lower variability in latency and lower box and whiskers show lower magnitude of latency. On each of the box plots, the box and whiskers representing latency performance are seen in pairs per priority level and the first box and whisker of each pair (red one) depicts the performance of a priority non-preemptive NoC based on Hermes and the second one (green one) depicts the performance of the NoC employing PFS.

1) Performance with random traffic

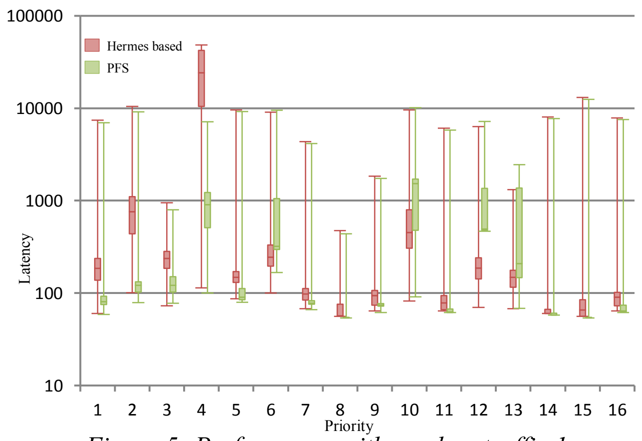


Figure 5: Performance with random traffic 1

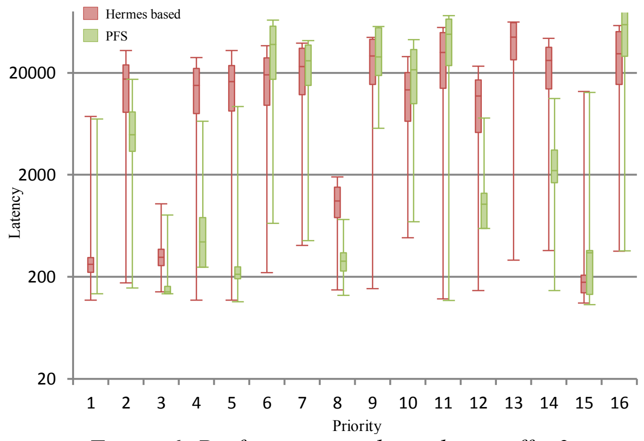


Figure 6: Performance with random traffic 2

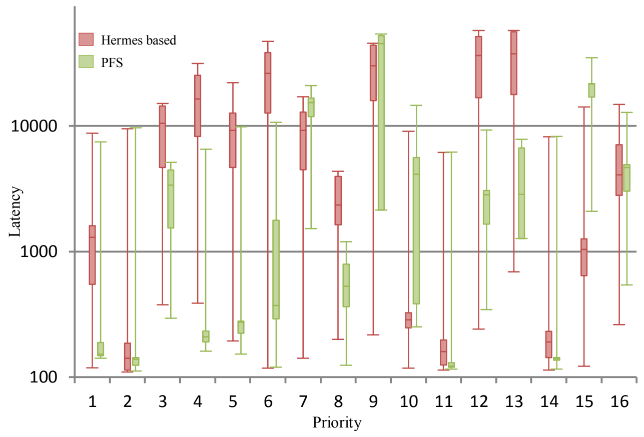


Figure 7: Performance with random traffic 3

To verify the performance merit of PFS, we tested the prototype with several random traffic scenarios (for 10^5 clock cycles) and the latency performance figures are interpreted as Figure 5, Figure 6 and Figure 7.

In the figures, it can be noted that the box and whiskers for the high priority packets (like 1 to 6) are seen mostly shorter and lower with PFS engaged compared to the basic NoC depicting lower variability and magnitude of latency. However, there are some packets that show higher magnitude or variability in latency like packet 6 in Figure 6 and packet 7 in Figure 7. These are side effects of PFS improving predictability of even higher priority packets and such effects occur as the characteristic of the specific traffic scenario used and the load level on the NoC.

2) Effect of load variation by varying payload flit count

To evaluate the effect of increased load due to the increase in the overall number of payload flits in the NoC, we tested a traffic scenario with increasing load level by varying the packet sizes. Test results with V values 0.7, 0.9, 1.3 and 1.5 can be seen as Figure 8, Figure 9, Figure 10 and Figure 11 respectively.

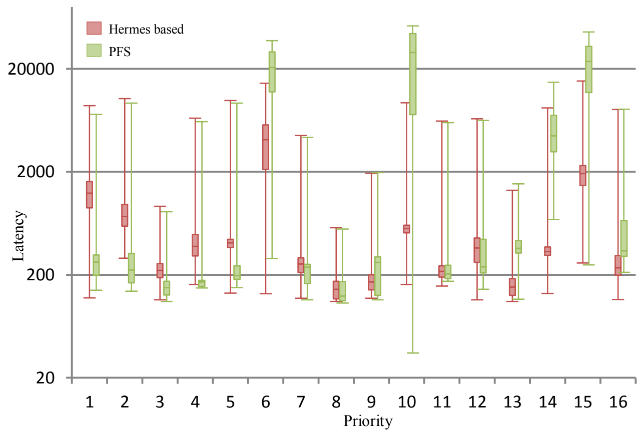


Figure 8: Performance at V=0.7

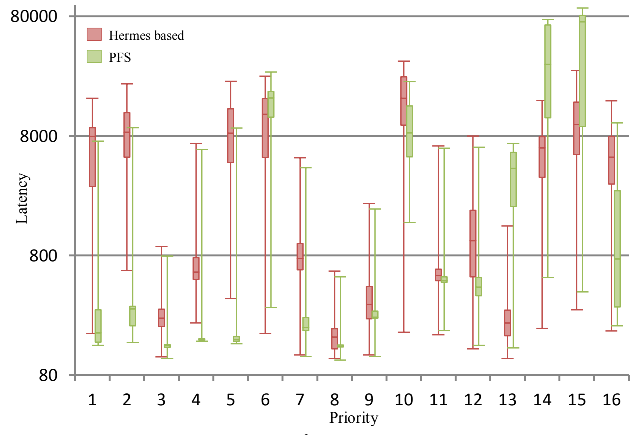


Figure 9: Performance at V=0.9

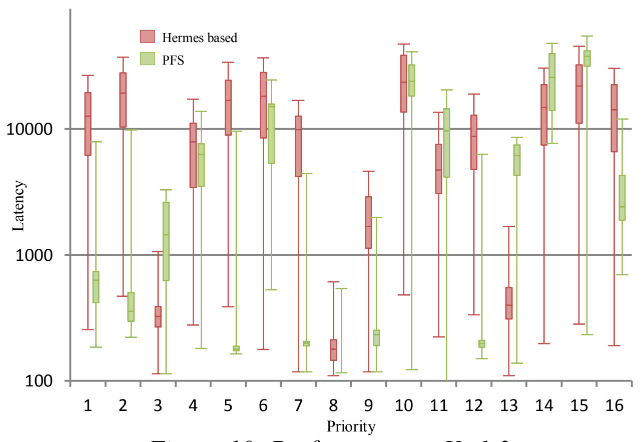


Figure 10: Performance at V=1.3

It can be seen in Figure 8, Figure 9 and Figure 10 that as the load on the NoC was increased, it had a negative impact on the latency performance thus resulting in longer and higher box and whiskers depicting higher variability and magnitude of latency.

With the basic NoC, this happens almost evenly irrespective of the priority value which can be unjustifiable with high priority packets. However with the employment of PFS, these effects are minimised for the high priority packets thus resulting in lower and shorter box and whiskers for high priority packets.

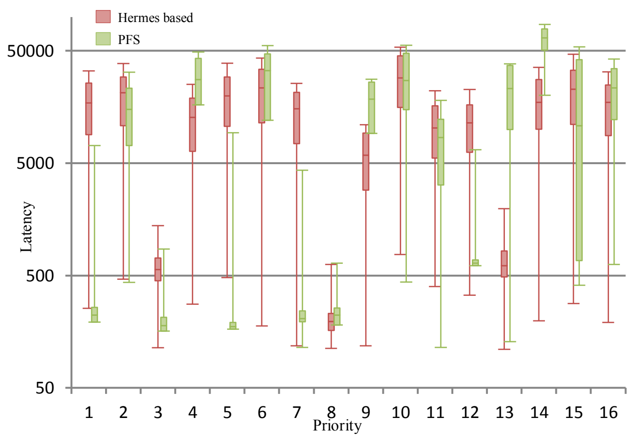


Figure 11: Performance at V=1.5

For the purpose of evaluation, we also tested the NoC with extremely high load of V = 1.5 and the test result show that the performance enhancement brought about to some high priority packets (like packets 1, 3 and 5) were severely affecting some lower priority packets (like packets 4 and 6) as evident from Figure 11.

Because packet 2 was sharing part of packet 1's traversal path as the characteristics of that specific traffic scenario, it can be seen that the performance enhancement brought about to packet 1 affects packet 2's latency negatively.

3) Effect of load variation by varying header flit count

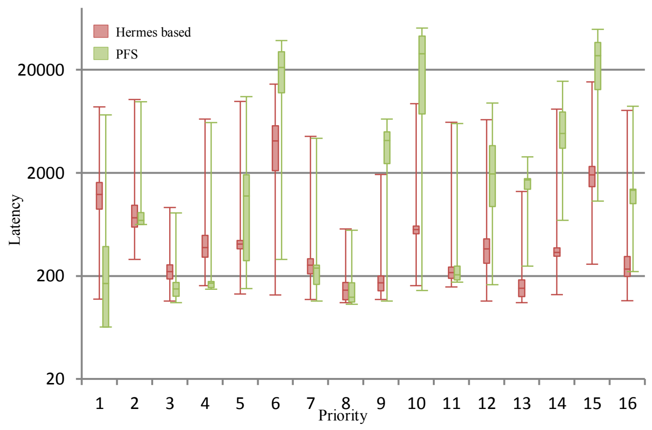


Figure 12: Performance at V=0.7

To evaluate the effect of increased load due to the increase in the overall number of header flits in the NoC, we tested a traffic scenario with increasing load level by varying the packet periods. Test results with V values 0.7, 0.9, 1.3 and 1.5 can be seen as Figure 12, Figure 13, Figure 14 and Figure 15 respectively. As seen with the tests in the previous section, the use of PFS was seen to cause lower latency variability and magnitude for the high priority packets on the cost of lower priority packets.

Test with extreme load of V=1.5 also seem to produce similar performance box plots as seen in the previous section.

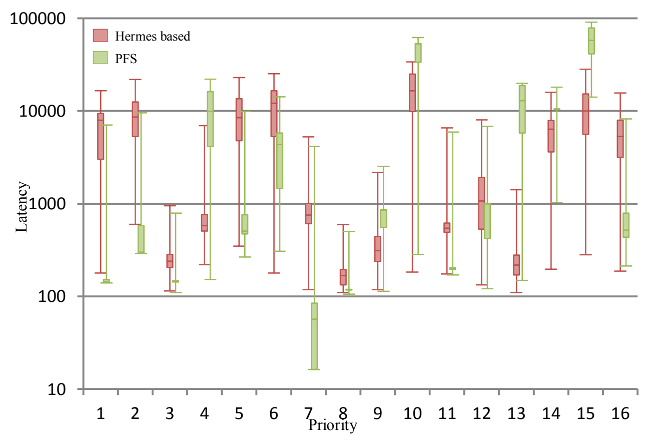


Figure 13: Performance at V=0.9

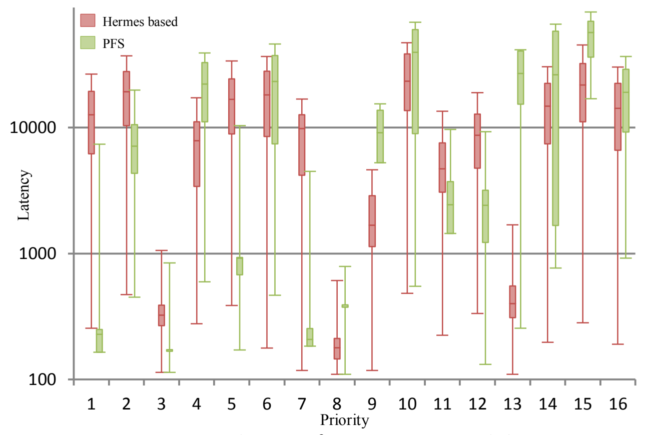


Figure 14: Performance at V=1.3

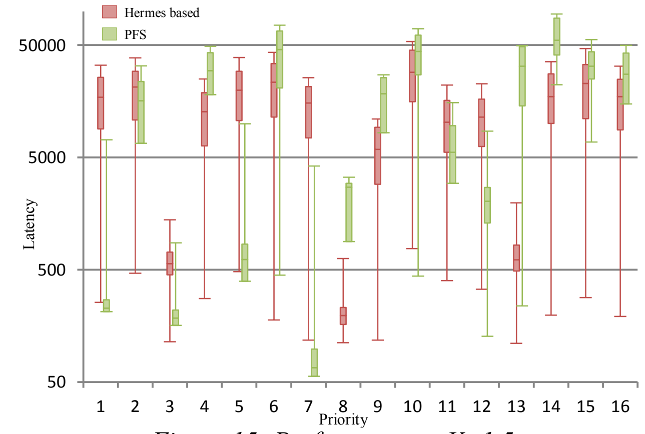


Figure 15: Performance at V=1.5

4) *Effect of RF and PD variation.*

To verify the effect of RF variation on performance, we tested a NoC with RF condition set to 1, ¾ and ½ of the packet size. With those tests the PFS logic was set to trigger only if the flits left to send owing to the lower priority was equal to or greater than 1, ¾ the packet size and ½ of the packet size respectively and the result is interpreted as Figure 16

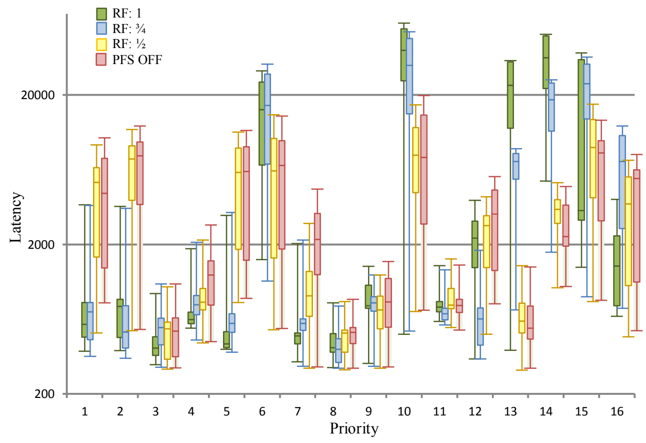


Figure 16: Effect of RF variation

In Figure 16, it can be observed that the effect of PFS can be scaled by varying the RF condition. With RF set to 1, the high priority packets were given the maximum preference and hence high priority packets are seen to be with the lowest variability and magnitude of latency. This comes at the cost of the low priority packet performance as those suffer higher variability and magnitude in latency. With the subsequent tests with RF set to ¾ and ½, the effect get moderated towards the low priority packet performance ultimately getting closer to the performance of the basic non-premptive NoC.

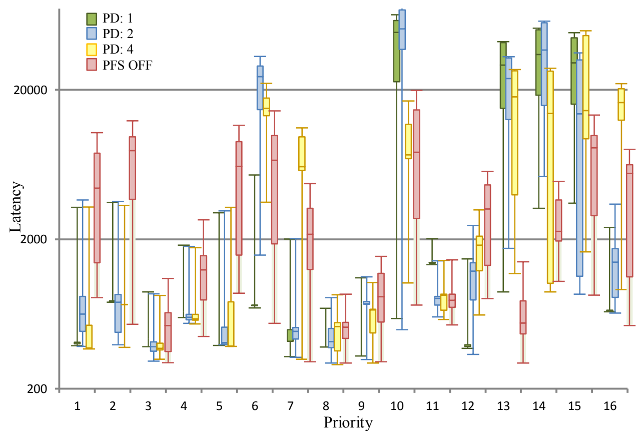


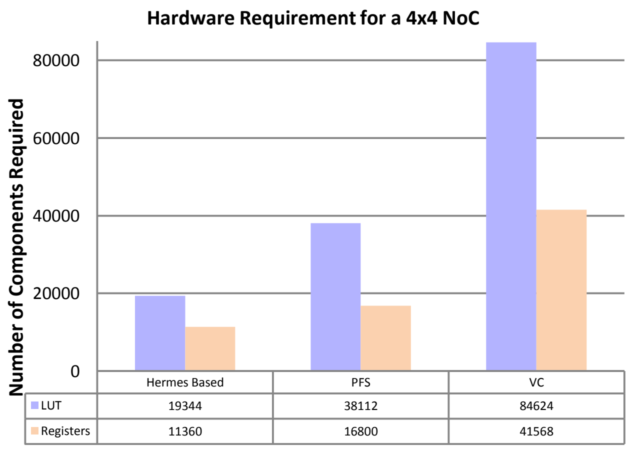
Figure 17: Effect of PD variation

As mentioned before, PD specifies the priority difference between competing flows and we conducted tests with PD set to 1, 2 and 4 and the result is interpreted as Figure 17. Though the variation of PF had similar effect as RF, it was seen to have wider and unpredictable performance variations than PD as evident from Figure 17. This is due to the fact that as PD scaling relays on the priority of the competing packets, it can get affected by the specific characteristics related to the task mapping used whereas RF is a more generic parameter. Due to this, RF can be deduced as a better parameter than PF for performance scaling applications.

B. Hardware Overhead

The hardware overhead associated with PFS was evaluated using Xilinx Vivado™ Design Suite by synthesising the prototype for a Xilinx Artix FPGA. While the baseline router (2-position input buffers) based on Hermes with priority arbitration utilised 1209 Look Up Tables (LUTs) and 710 of Slice Registers of the chosen FPGA, the PFS enabled NoC (2-position input buffers) utilised 2382 LUTs and 1050 Slice registers which is minimalistic.

As seen with [13] in Section II, Virtual Channel implementation is expensive and when a comparable Virtual Channel (4 VCs with 2-position buffers) based design we implemented upon the same basic NoC architecture was tested, Vivado generated a LUT requirement of 5289 and Slice Register requirement of 2598 thereby demonstrating the overhead associated.



	Hermes Based	PFS	VC
LUT	19344	38112	84624
Registers	11360	16800	41568

Figure 18: Hardware overhead comparison

From the results, the hardware overhead comparison for the techniques on a 4x4 NoC is interpreted in Figure 18. It can be observed that PFS has moderate hardware requirements compared to the basic non-preemptive NoC while the Virtual Channel based NoC costs more than four times.

VI. Conclusion

This paper has introduced the PFS technique by which predictability of non-preemptive NoCs can be enhanced without causing major hardware overheads. PFS employs a combination of priority forwarding and selective packet splitting technique to resolve HOL blocking and tail backing of NoC packets thereby improving packet predictability. The effectiveness of the technique was evaluated using a prototype and it was tested with different traffic patterns and load levels and the use of PFS was found to reduce variability and magnitude of latency of high priority packets with the hardware overhead of 97% LUTs and 47% registers.

VII. Acknowledgement

This work is supported by EPSRC project LowPowNoC (EP/J003662/1) and EU FP7 projects T-CREST (288008) and DREAMCLOUD (611411).

VIII. References

[1] W. J. Dally, 'Virtual-channel flow control', in *Proceedings of the 17th annual international symposium on Computer Architecture*, New York, NY, USA, 1990, pp. 60–68.

[2] R. Stefan, A. Molnos, A. Ambrose, and K. Goossens, 'A TDM NoC supporting QoS, multicast, and fast connection set-up', in *Design, Automation Test in Europe Conference Exhibition (DATE), 2012*, 2012, pp. 1283 –1288.

[3] L. Thiele and R. Wilhelm, 'Design for Timing Predictability', *Real-Time Systems*, vol. 28, no. 2–3, pp. 157–177, Nov. 2004.

[4] T.-C. Huang, U. Y. Ogras, and R. Marculescu, 'Virtual Channels Planning for Networks-on-Chip', in *8th International Symposium on Quality Electronic Design, 2007. ISQED*, 2007, pp. 879–884.

[5] B. Sudev and L. S. Indrusiak, 'PFT- A low overhead predictability enhancement technique for non-preemptive NoCs', in *2013 IFIP/IEEE 21st International Conference on Very Large Scale Integration (VLSI-SoC)*, 2013, pp. 314–317.

[6] B. Sudev and L. S. Indrusiak, 'Predictability Enhancement in Non-preemptive NoCs using Selective Packet Splitting'. 12th IEEE International Conference on Industrial Informatics (INDIN), Porto Alegre-Brazil, Jul-2014 *(Accepted)*.

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

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

[9] R. Dobkin, R. Ginosar, and I. Cidon, 'QNoC Asynchronous Router with Dynamic Virtual Channel Allocation', in *1st International Symposium on Networks-on-Chip,.2007*, 7, p. 218.

[10] T. Bjerregaard and J. Sparso, 'A router architecture for connection-oriented service guarantees in the MANGO clockless network-on-chip', in *Design, Automation and Test in Europe, 2005. Proceedings*, 2005, pp. 1226 – 1231 Vol. 2.

[11] S. Park, T. Krishna, C.-H. Chen, B. Daya, A. Chandrakasan, and L.-S. Peh, 'Approaching the theoretical limits of a mesh NoC with a 16-node chip prototype in 45nm SOI', in *2012 49th ACM/EDAC/IEEE Design Automation Conference (DAC)*, 2012, pp. 398–405.

[12] A. Kahng, B. Li, L.-S. Peh, and K. Samadi, 'ORION 2.0: A fast and accurate NoC power and area model for early-stage design space exploration', in *Design, Automation & Test in Europe Conference & Exhibition, 2009. DATE '09.*, 2009, pp. 423–428.

[13] 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 18th annual symposium on Integrated circuits and system design*, New York, NY, USA, 2005, pp. 178–183.

[14] A. Morgenshtein, A. Kolodny, and R. Ginosar, 'Link Division Multiplexing (LDM) for Network-on-Chip Links', in *2006 IEEE 24th Convention of Electrical and Electronics Engineers in Israel*, 2006, pp. 245 –249.

[15] P. Kundu, 'On-Die Interconnects for Next Generation CMPs'. Proc. Workshop On- and Off-Chip Interconnection Networks for Multicore Systems,, Dec-2006.

[16] 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.

[17] R. Manevich, I. Cidon, A. Kolodny, I. Walter, and S. Wimer, 'A Cost Effective Centralized Adaptive Routing for Networks-on-Chip', in *Digital System Design (DSD), 2011 14th Euromicro Conference on*, 2011, pp. 39 –46.

[18] 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.