Network-on-Chip Packet Prioritisation based on Instantaneous Slack Awareness

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

***Abstract*— Arbitration policies and predictability enhancement measures typically employ packet priority as the decisive parameter. Though packet timeliness is a key attribute, Network-on-Chip designs rarely consider timeliness as a parameter mostly due to the impracticality of utilising time stamping which relay on the notion of a global time. In this paper, we introduce a low overhead approach where packets carry a slack value, which would notify the router of the latency the packet can suffer without any adverse effects. This would enable routers to service late packets (even lower priority ones) by trading the expendable time associated with the high priority packets hence improving overall quality of service. Utilising a Hardware Description Language coded prototype, we demonstrate the effectiveness of the technique and quantify the associated hardware overhead.**

Keywords- Network-on-Chip, Predictability, Timeliness, On-chip networks, Arbitration, Prioritisation

I. INTRODUCTION

In multicore and many-core systems, on-chip communication has been identified as a performance bottleneck. Networks-on-chip (NoCs) have been widely proposed as a standardised and scalable network platform capable of transporting application traffic, and providing performance guarantees such as time predictability [1]. In real-time systems in which the application structure and system workload is known ahead of time, static analysis can be used to determine suitable packet priorities and mappings [2]. However, in open applications the workload which the platform must handle can be unknown at design time. This can be because tasks or data flows may arrive dynamically requesting immediate transmission, but nevertheless requiring a certain quality of service (QoS). Alternatively, in a heterogeneous architecture, known applications may have to coexist with dynamically admitted traffic. These situations require additional flexibility in arbitration decisions beyond static priorities, making routers aware of the timeliness of in-progress packets.

This paper proposes DHARA (Dynamic slack Hard-line Aware Router Architecture) a protocol in which arbitration decisions are made on the basis of a dynamically computed priority value. Packet headers are augmented with an additional slack value, which represents the latency that the packet can endure to its destination without adverse effects. This slack value is decremented by intermediate arbiters while the packet is blocked and forced to wait. During arbitration decisions, an instantaneous priority is computed from this slack value and the application-supplied priority value. This dynamic priority adjustment allows lower priority packets which have been waiting for longer to be serviced, while trading off some expendable time on early high priority packets.

The paper specifies DHARA and demonstrates its benefits by evaluation using a hardware prototype implemented in BlueSpec System Verilog, using synthetic traffic together with a case study of a real NoC application augmented with additional synthetic traffic. DHARA is evaluated combined with our earlier work called Priority Forwarded Packet Splitting (PFS) [3], which is designed to improve packet predictability of dynamic traffic. The hardware overheads of DHARA are quantified and found to be reasonable in terms of the reduction in packet lateness provided.

II. BACKGROUND

With factors like scalability and performance limiting the employment of buses in large many-cores, there have been several advancements in NoC designs over the years. With a variety of tuneable parameters available on NoCs, several architectures have been developed aiming at different performance requirements and limitations in overhead. With NoCs like Hermes[1], the hardware overhead was minimalistic but the advantage came at the price of uncertainty in packet latencies. Time Division Multiplexing (TDM) [4] solved the unpredictability altogether, however it came at the cost of limited scalability and restricted dynamic behaviour. Virtual Channels (VC) [5] provided a more dynamic infrastructure than TDM with better predictability than non-preemptive NoCs. This was achieved by splitting communication into service levels and then by providing separate logical channels for each service level. However this resulted in high hardware overheads as seen in [6] where the hardware overhead was seen to linearly increase with the number of VCs.

Our previous work involved improving predictability [7] in NoCs by developing scalable dynamic techniques that alter router behaviour. With [3], [8] and [9], we employed dynamic techniques that can neutralise Head-of-Line (HOL) blocking and tailbacking thus improving packet predictability.

All of the above techniques consider the application-supplied priority as the sole decisive parameter. Thus, the routers favour high priority packets over low priority packets

978-1-4799-6648-6/15/$31.00 ©2015 IEEE

while implementing arbitration, preemption or other predictability enhancement measures despite its timeliness. As timeliness of the packet can be a critical issue to consider, Das et al in [10] presented a slack aware system where the packet header would include the priority value which constituted of both its packet priority and acceptable slack. The slack value was static and was based on parameters like number of hops or maximum latency level. This does not however take into account the time spent by packets waiting in NoC routers for arbitration. To compensate for that, the paper employs multiple network interface queues and batching which is hardware expensive. Andreasson et al in [11] presented an approach which relied on using slack (or unused slots) on TDM based systems. With this approach, the TDM based functionality of the router made the notion of timeliness in packets unnecessary but as with the classical TDM approach, it limited its scalability and dynamic behaviour. Similarly, Diemer et al in [12] depicted a back suction based flow control which was used to improve Best Effort service latency by utilising the free bandwidth available with their Guaranteed Service infrastructure. Berejuck et al in [13] tried to improve QoS in VC based NoCs by targeting ageing of packets. In the work, the packets were added with fields in their headers that would be incremented as packets wait for arbitration. This value is then utilised by the arbitrator for arbitrating packets of the same VC. Similarly, Correa et al in [14] presents a NoC framework that allows the routers to increase packet priority when a packet waits for arbitration for certain number of clock cycles. However, under high load condition, there is possibility of multiple packets acquiring highest priorities thus compromising the predictability of the high priority spectrum of packets. As seen with [15], the typical method to introduce the notion of time is by time stamping. Time stamping relies on the access to a global time thus requiring long counters which is impractical in NoC routers. With DHARA, we introduce a novel approach by which the packets carry information that would notify the routers of the residual slack of the packet, thus providing the opportunity to provide preference to packets that are late in time. As the system does not rely on a global time; the hardware requirement for the system in minor thus improving its practicality.

III. DHARA BASED SLACK AWARENESS

The means employed to utilise the residual slack (which is the notion of time) depends on the type of predictability enhancement measures available in that NoC. In a simple NoC system, evaluating the timeliness of the packet can enable the router to prioritise a late low priority packet over an early high priority packet (one with residual slack) by trading the time the high priority packet considers expendable. This would improve the latency of the low priority packet without any side effects to the associated higher priority packet. With DHARA, we enable the IP or the Network Interface to provide an additional parameter to packets (apart from priority and destination); the slack or delay the packet can endure without any adverse effects. Each packet would carry an additional field in the header which would hold the slack information and this is treated similarly to the priority information in the header. So every time a packet header is injected into an input port, the slack gets stored into a register. If the packet gets arbitration immediately, slack along with the rest of the parameters would be forwarded to the next router as they are embedded in the packet header.

On the other hand, if the packet is delayed, the value inside the slack register would get decremented every time a slack-interrupt is generated in the router. To enable the router to decrement the data inside slack registers during contention, the routers are augmented with an incrementing counter that would produce an interrupt called slack-interrupt every time it overflows. The slack-interrupt generator has an adjustable scale pointer using which the granularity of lateness can be varied. For example, if the scale pointer is set at zero, the system would provide an interrupt every two clock cycles and so the slack value would be decremented every two clock cycles the packet is forced to wait. The granularity would be equal to $2^{\text{`scale pointer value'}+1}$ and hence if the pointer is set to 7 (as an example), the value inside the slack register would be decremented every $2^8 = 512$ clock cycles the packet is forced to wait. With our current work, this scale pointer value is static and is set during design time. In future, we plan to make the slack pointer value dynamic by embedding it into the packet header. As DHARA does not require access to a global time, the hardware requirement is relatively low thus enhancing its practicality.

IV. IMPLEMENTATION

A. Basic prototype architecture

The NoC prototype was designed based on Hermes hence following a five-port architecture and to reduce hardware requirements, it employed XY-routing and wormhole switching. Used in a uniform mesh topology, packet headers are added with priority fields (to carry application supplied priory value) that are utilised by arbitration units in routers to resolve contention between packets over output ports. The routers are designed with buffered input ports and on reception of a packet header; they employ the routing logic to set the 'port request' register and the 'priority' register (inside the respective input port) after evaluating the information carried in the header. The arbitration unit in the router evaluates 'port request' and 'priority' register values inside all input ports to provide arbitration by setting the 'out port' register inside the respective input port. This configures the input port to send flits to the local IP or the neighbouring router through the allocated output port. The router then continue flit transfer until it encounters a tail flit so that once it is detected, the connection can be closed by resetting the 'out port' register.

In this paper, we use routers that employ the predictability enhancement technique Priority Forwarded Packet Splitting (PFS) [3] to evaluate the performance advantages brought about with DHARA. PFS aims to reduce latency of higher priority packets at the cost of lower priority packet latencies employing two techniques; Selective Packet Splitting (SPS) [9] and Priority Forwarding [8]. SPS aims at resolving tailbacking of high priority packets by low priority packets by employing a

low overhead version of preemption by which the low priority packet will be split by adding a tail flit. To enable SPS, the router's state machine has additional logic to end a communication if it detects a higher priority packet requiring arbitration to the same link. This is done by sending a tail flit as part of the current communication thereby automatically ending the communication downstream routers. The router then issues a new arbitration request to the splitted packet so that once the high priority communication is transmitted; the splitted communication will be resumed. On the other hand, Head of Line (HOL) blocking caused by a blocked low priority packet blocking a high priority packet is resolved by employing Priority Forwarding. To enable this, the router has logic to detect any blocked low priority packets blocking high priority packets. Under such situations, the priority information (of the high priority packet) would be send down the line using dedicated links so that the blocking issue can be resolved by temporarily boosting the priority of the low priority blocked packet's arbitration request.

B. DHARA based slack awareness

As a starting point, we used a PFS enabled router and modified it to encompass slack awareness. Logic was added to packet generators to add slack values to headers and the slack value was set at seven bits. The highest value possible (i.e. 127) was treated as packets with the notion of timeliness disabled (where PFS would never be enabled). With slack values less than 127, the routers would decrement the value if the packet is detained for a number of clock cycles determined by the scale pointer. As the slack value (which is the notion of earliness) would be decremented only when the packet is in the front of the FIFO buffer (thereby initiating an arbitration request), there is possibility for the packet header to be behind other flits, thus waiting time unaccounted for. To resolve such HOL blocking of headers inside buffers, we modified the buffers so that every time a flit is injected into the buffer, the newly added logic will verify whether it is a header and if it is; the slack register would be updated. As this happens before the packet gets to the front of the queue, the routers would be able to decrement the slack value irrespective of the position of the header in the FIFO and hence provide a more reliable awareness of slack. In the current prototype, all computational units including the arbiter, Priority Forwarding logic and Packet Splitting logic work based on instantaneous priority rather than the priority information in the packet header. The instantaneous priority is estimated using Equation 1 which employ an addition and a right shift (>>) operation thus enabling efficient realisation in hardware.

$$P_I = P_P + (S \gg D) \qquad (1)$$

(P_I – Instantaneous priority, P_P – Packet priority, S – Slack value, D- Divider index)

As seen in the equation, the instantaneous priority is estimated by summing the packet priority with the slack value shifted to the right D number of times. Practically, D can be set to 0, 1 or 2 hence realising S, S/2 and S/4 respectively thus varying the weightage of the timing parameter in the equation (with routers favouring packets with lower magnitude of P_I under contention).

C. Performance analysis framework

The evaluation framework was coded in Bluespec System Verilog as a router design enveloped in a parameterisable test bench. The test bench was used to replicate routers and interconnect them on a 2D-mesh topology. The local ports of routers were connected to packet generator modules that can be pre-set with packet parameters like priority, packet size and destination. The packet generator configuration data is designed to be auto generated as Bluespec source code using a custom built code generator which could configure them randomly or as per a series of algorithms so as to generate specific traffic patterns. On packet generation or reception, the occurrence is documented as an entry onto the data file in order to allow our custom built macro code (running inside spread sheet software) to analyse it and generate latency statistics and graphs.

$$V = \left\{\sum_{x=0\ to\ W-1} \sum^{y=0\ to\ H-1} \left(\frac{D_{x,y}}{P_{x,y}}\right)\right\} / L_{(W \times H)} \qquad (2)$$

(W- NoC Width, H- NoC Height, D- No load latency, P- Period of computation, L- Links)

The load on the NoC (V) is estimated using equation 2 and a 4x4 NoC was simulated and the performance was analysed with diverse traffic patterns and load levels.

V. IMPLEMENTATION RESULT

A. Performance with random traffic

The latency performance of the technique is typically 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 show the extreme cases of latency and the boxes indicate the upper and lower quartile of latency with the middle line depicting the median. Therefore, shorter box and whiskers show lower variability in latency and lower box and whiskers show lower magnitude of latency. The paper also include average latency plots with similar X and Y axis parameters. To evaluate the system performance, we tested the prototype with random traffic scenarios with load approximately at V=0.6. The latency performance of prototype interpreted as box plots is added as Figure 1(a) and Figure 2(a). In the two figures, the box and whiskers are seen as triples with first one representing the performance of a simple Hermes based NoC with packet priority (designated H_p), the second one representing a PFS based router and the third one representing the DHARA based slack aware PFS router (designated PFS-D). In Figure 1(a) and Figure 2(a), the improvements brought by PFS is quite clear as high priority packets (1 to 7) are seen to suffer lower variation and magnitude of latency depicted by the shorter and lower box and whiskers than the basic H_p NoC. However, the low priority packets are seen to have higher variations and magnitudes of latency depicted by the longer and higher box and whiskers. With the PFS-D tests, the system was configured to provide all packets with a slack of 20 with scale pointer set at 7 and divider index at 0. As a result, it can be seen that the higher priority packets latencies are boosted thus marginally improving the latency performance of low priority packets. The average latency performance of the three traffic scenarios can be seen as Figure 1(b) and Figure 2(b). Similar to the box

plots, it can be seen that with PFS the average latency of the high priority packets (1 to 7) are very low comparted to H_p. As this advantage comes at the cost of low priority packet latencies, they are seen to have high magnitudes. Similar to what was seen with the box plots, the DHARA based slack aware PFS is seen to moderate the worst case performance of the low priority communication trading the expendable slack available with the higher priority packets.

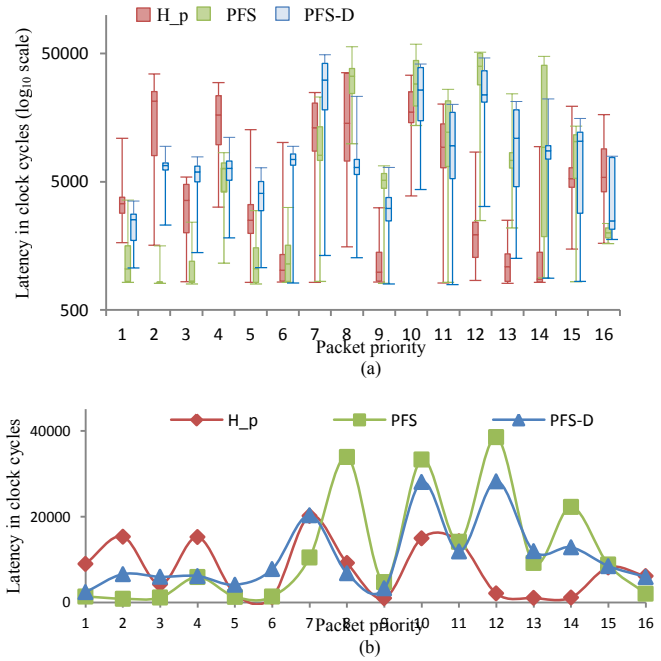

Figure 1: Performance with random traffic 1 (a) Latency (b) Average Latency

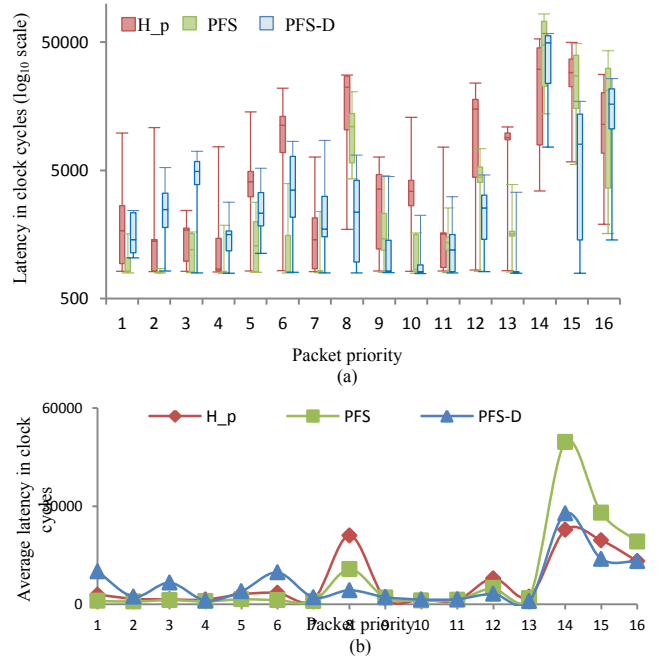

Figure 2: Performance with random traffic 2 (a) Latency (b) Average Latency

Figure 3 and Figure 4 shows the cumulative frequency plot of the number of late packets with random traffic 1 and 2 respectively. The plots show the cumulative frequency of packets that had lateness of more than its expected latency (basic latency plus slack). As evident from the plots, the H_p NoC features late high priority packets (like priority 3 in both the figures) while the PFS based NoC encounter lateness only with low priority packets (like packets 11 to 16 in Figure 3 and packets 9 to 11 in Figure 4). As a result of the severity of the approach, the low priority packets are seen to suffer increased number of late packets. With DHARA, this effect is moderated by trading the slack from higher priority packets as evident in the plots.

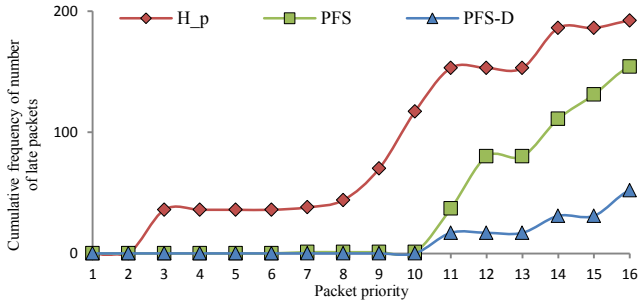

Figure 3: Cumulative frequency of number of late packets (random traffic 1)

In Figure 3, DHARA is seen to trade the slack from high priority packets to such an extent that the total number of late packets are less than 50% of the other two approaches.

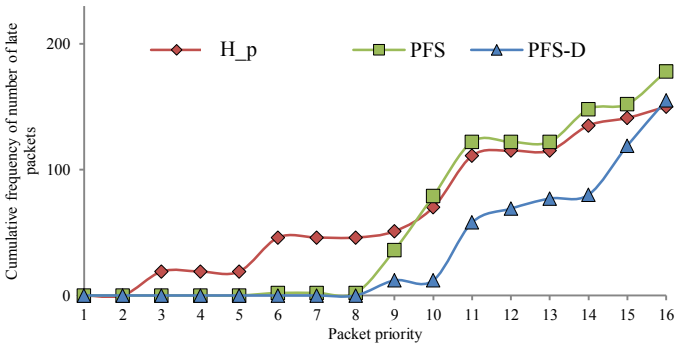

Figure 4: Cumulative frequency of number of late packets (random traffic 2)

With random traffic 2 however, it can be seen that the total number of late packets with DHARA is similar to the H_p NoC but the bulk of this is attributed to packets with the lowest priority values (packets 11, 15 and 16) which is acceptable.

B. Performance with load variation

To evaluate the performance of the system with varying load, we used random traffic 2 as the base line and tested it with varying load levels. The average latency plots of random traffic 2 with V=0.67, V= 0.83 and V=1.1 can be seen as Figure 5, Figure 6 and Figure 7 respectively.

In Figure 5, Figure 6 and Figure 7, it can be seen that the H_p NoC suffers random peaks in average latency despite the packet priority. With PFS it can be seen that the high priority

packets suffer low average latency while the low priority packets suffer a few peaks. With the timing awareness turned on, it can be seen that the high peaks seen with the low priority packets (with PFS approach) were moderated by the routers by delaying the high priority packets moderately thus improving overall QoS.

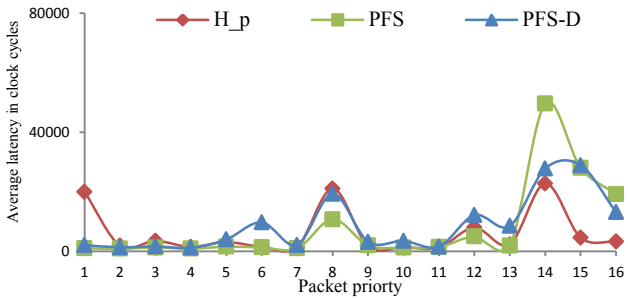

Figure 5: Average latency at V=0.67

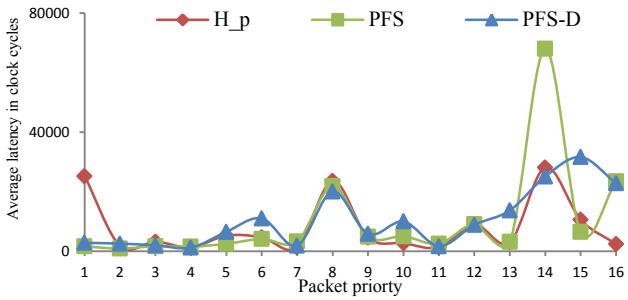

Figure 6: Average latency at V=0.83

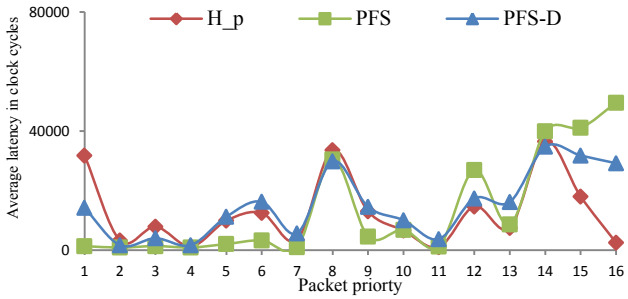

Figure 7: Average latency at V=1.1

This phenomenon is more evident in Figure 8, Figure 9 and Figure 10 where the maximum latency plots of H_p NoC, PFS based NoC and PFS-D based NoC respectively are added at varying load levels.

In Figure 8, it can be seen that with the increase in load, the latency levels gets high peaks despite packet priority level. For example, even with the highest priority, packet 1 is seen to have increased maximum latency peaks with the increase in load.

As seen in Figure 9, despite the increase in load, the PFS based NoC suffers minimal variation in maximum latency for high priority packets (1 to 7) while low priority packets suffer major variations. By employing timing awareness in PFS routers (Figure 10), the high peaks of the low priority packets were moderated trading expendable time from the higher priority packets. This did result in minor increase in maximum latency of high priority packets which was at moderate levels. The three lines depicting each load levels are also seen to be following the same pattern and are seen to be close to each other, depicting minor variation in maximum latency despite increase in load.

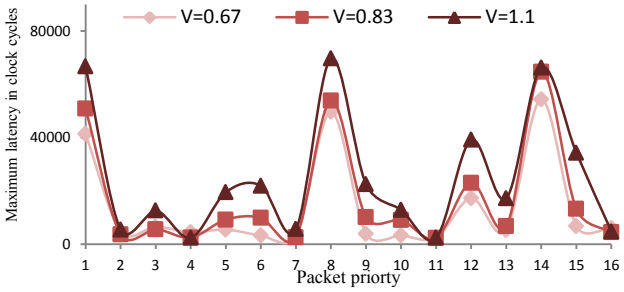

Figure 8: Maximum latency of H_p NoC packets with load variation

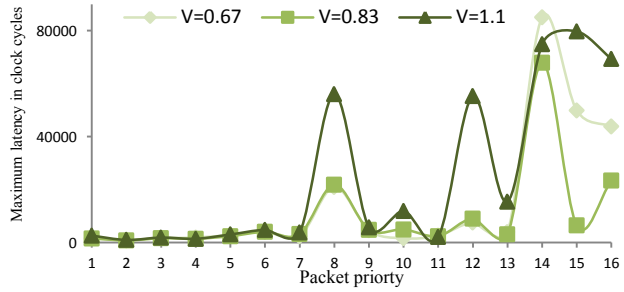

Figure 9: Maximum latency of PFS based NoC packets with load variation

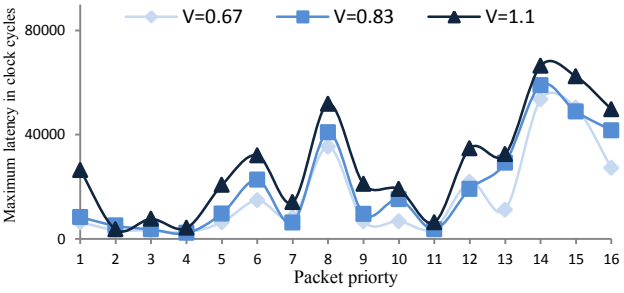

Figure 10: Maximum latency of PFS-D based NoC packets with load variation

C. Performance with realistic task mapping

We tested the system with a traffic scenario based on the application used in [2] and Figure 11 shows the cumulative frequency distribution of late packets under the scenario. For this experiment, we also tested a classical Hermes router based NoC (without any packet prioritisation) designated as H. It can be seen that the H NoC packets suffer high magnitudes of late high priority packets. The addition of packet priorities does improve the situation marginally as evident from H_p's performance plot. This scenario is improved very much with PFS and by using DHARA, the number of late packets are seen to decrease even further.

To evaluate the effect of additional packet flows, we added ten more packet flows (with lowest priorities) scaling the packet periods to keep the load in similar levels. As seen in Figure 12,

the results prove comparable to the previous experiment however; a few of the lowest priority packets (packets 40 to 42) are seen to have increased number of late packets with DHARA. With all of the tests in the paper, we assigned equal slack value to all the packets however; the efficiency of the system can be improved further by customising slack allocation per packet depending on requirements.

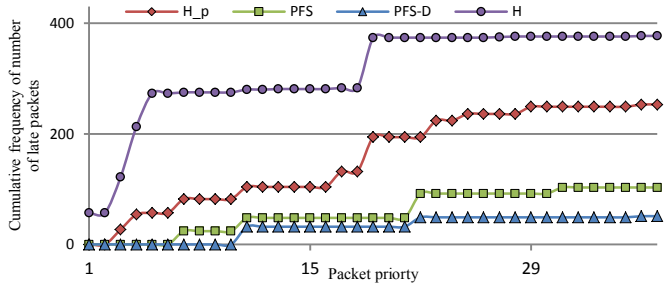

Figure 11: Cumulative frequency of number of late packets

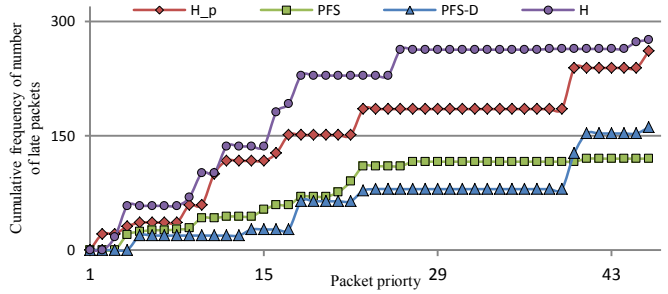

Figure 12: Cumulative frequency of number of late packets with increased number of packets.

D. Hardware overhead

The hardware requirements for a PFS based router (2382 LUTs and 1050 registers) compared to PFS-D (2763 LUTs and 1176 registers) design was evaluated using Xilinx Vivado and was found to be minimalistic with only 16% more lookup tables and 12 % more registers on a .

VI. Future work

Though preliminary tests revealed that the system performance can be varied by changing scale pointer and divider index values, future work will involve testing the characteristics of the performance variations resulting from the parameter tuning in detail.

The Next generation of DHARA would also look into making the scale pointer dynamic by embedding it into the packet header. This would increase the precession of slack assignment and could improve the performance further.

Previously, we were involved in a work that extended the classical Virtual Channel approach called Dynamic Time Multiplexed Virtual Channels (DTMVC) [16] where a VC based router would be able to switch its performance settings depending on requirements. With DHARA based slack awareness, DTMVC routers would be able to switch the performance settings depending on the slack available on high priority service level packets thus becoming a self-regulating closed loop system.

VII. Conclusion

This paper introduced a system in which the notion of packet timeliness can be provided to the routers thus enabling them with an additional metric for dealing with contention between packets. The system enabled the packet header to carry a residual slack value which denoted the lateness the packet can endure without adverse effects. As packets would get blocked, the slack value would get decremented thus updating the lateness level. Extensive testing using a slack aware version of a PFS technique enabled prototype showed improvement in low priority packet latency by trading expendable time higher priority packets had. Tests carried out with synthetic load as well as with realistic application prove notable reduction in late packet numbers. As the system did not require access to global time to evaluate timing, the hardware overhead for timing awareness was seen to be minimalistic requiring just 16% more lookup tables and 12 % more registers.

VIII. Acknowledgement

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

IX. References

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

[2] Z. Shi and A. Burns, “Schedulability Analysis and Task Mapping for Real-time On-chip Communication,” *Real-Time Syst.*, vol. 46, no. 3, pp. 360–385, Dec. 2010.

[3] B. Sudev and L. S. Indrusiak, “Low overhead predictability enhancement in non-preemptive network-on-chip routers using Priority Forwarded Packet Splitting,” *ReCoSoC*, 2014, pp. 1–8.

[4] R. Stefan, A. Molnos, A. Ambrose, and K. Goossens, “A TDM NoC supporting QoS, multicast, and fast connection set-up,” *DATE,* 2012, pp. 1283 –1288.

[5] W. J. Dally, “Virtual-channel flow control,” *ISCA*, New York, NY, USA, 1990, pp. 60–68.

[6] A. Mello, L. Tedesco, N. Calazans, and F. Moraes, “Virtual channels in networks on chip: implementation and evaluation on hermes NoC,” *ICSD*, New York, NY, USA, 2005, pp. 178–183.

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

[8] B. Sudev and L. S. Indrusiak, “PFT- A low overhead predictability enhancement technique for non-preemptive NoCs,” in *VLSI-SoC*, 2013, pp. 314–317.

[9] B. Sudev and L. S. Indrusiak, “Predictability Enhancement in Non-preemptive NoCs using Selective Packet Splitting.” *INDIN*, Porto Alegre-Brazil, Jul-2014.

[10] R. Das, O. Mutlu, T. Moscibroda, and C. Das, *Aérgia: Exploiting Packet Latency Slack in On-Chip Networks*. *SCA* 2010, France.

[11] D. Andreasson and S. Kumar, “Slack-time aware routing in NoC systems,” *ISCAS*, 2005, pp. 2353–2356 Vol. 3.

[12] J. Diemer and R. Ernst, “Back Suction: Service Guarantees for Latency-Sensitive On-chip Networks,” *NOCS*, 2010, pp. 155–162.

[13] C. A. Z. Marcelo Daniel Berejuck, “Adding mechanisms for QoS to a network-on-chip.,” *SBCCI*, 2009.

[14] E. de F. Corrêa, L. A. de P. e Silva, F. R. Wagner, and L. Carro, “Fitting the Router Characteristics in NoCs to Meet QoS Requirements,” *ICSD*, New York, NY, USA, 2007, pp. 105–110.

[15] C. Ciordas, T. Basten, A. Radulescu, K. Goossens, and J. Meerbergen, “An event-based network-on-chip monitoring service,” HLDVT, 2004,

[16] B. Sudev and L. S. Indrusiak, “Dynamic Time Multiplexed Virtual Channels, a Performance Scalable Approach in Network-On-Chip Routers to Reduce Packet Starvation,” *YDS*, 2014, pp. 21–30.