DeBAR: Deflection Based Adaptive Router With Minimal Buffering

John Jose, Bhawna Nayak, Kranthi Kumar and Madhu Mutyam
PACE Laboratory, Department of Computer Science and Engineering
Indian Institute of Technology Madras, Chennai, India 600036
{johnjose, bhawnan, kranthi, madhu}@cse.iitm.ac.in

***Abstract*—Energy efficiency of the underlying communication framework plays a major role in the performance of multicore systems. NoCs with buffer-less routing are gaining popularity due to simplicity in the router design, low power consumption, and load balancing capacity. With minimal number of buffers, deflection routers evenly distribute the traffic across links. In this paper, we propose an adaptive deflection router, DeBAR, that uses a minimal set of central buffers to accommodate a fraction of mis-routed flits. DeBAR incorporates a hybrid flit ejection mechanism that gives the effect of dual ejection with a single ejection port, an innovative adaptive routing algorithm, and a selective flit buffering based on flit marking. Our proposed router design reduces the average flit latency and the deflection rate, and improves the throughput with respect to the existing minimally buffered deflection routers without any change in the critical path.**

I. Introduction

Rapid increase in the number of on-chip processing cores has increased the importance of effective and scalable communication framework. Scalable packet switched Network-on-Chips (NoCs) are developed to serve the growing communication needs of such large systems. Router micro-architecture plays a vital role in the performance of an NoC system. Performance fine tuning in terms of packet latency, and network power makes the NoC router design a critical challenging task. Any competitive router design should focus on a short critical path with minimal control logic and buffer footprint [1]. Input buffered Virtual Channel (VC) routers [2] dominated the first generation NoC designs due to their simple wormhole switching [3] and high load handling capacity. But they consume large portion of chip power due to the presence of buffers. Studies show that approximately 30% to 40% of chip power is consumed by the NoC [4], [5].

Buffer-less routers are proposed to tackle the rising power concerns associated with the traditional VC routers. Considering the packet injection behavior of the real workloads, the input buffers in VC routers are over provisioned [1]. Our simulations with the real workloads on mesh NoCs with VC routers show that for low-injection rate applications, in 90% cases, less than 25% of the buffers are being occupied, thereby exposing the over provisioning of buffers in VC routers. For low to medium injection rate applications buffer-less NoC router design is an optimal design choice.

We propose a novel **De**flection **B**ased **A**daptive **R**outer (DeBAR), with a minimal central buffer pool that stores a fraction of the misrouted flits. DeBAR dynamically controls the flit injection from local core and the central buffer pool to ensure fairness. Experiments on 8x8 mesh with synthetic traffic patterns [3] and SPEC CPU-2006 benchmark mixes [6] show that DeBAR outperforms the existing best baseline minimally buffered deflection router MinBD [7] in terms of the average flit latency, deflection rate, and throughput.

II. Buffer-less Routers: Related Work

Traditional VC routers have buffers in their input ports. Flits reside in the VC buffers until a productive output port is obtained. As buffers in the NoC routers are power hungry and buffer management circuits are complex, buffer-less routers have gained popularity [1], [8], [9]. When a flit reaches a buffer-less router, if its desired output port is not available, the flit is either dropped [10], [9] or deflected to an undesired port [1], [8], [7]. In the former case, the dropped flit is retransmit from the source using necessary acknowledgment mechanism. Overheads involved in coordinating the acknowledgments and retransmissions reduced the popularity of the dropping models. In the later case, the deflected flits eventually reach the destination by proper livelock prevention mechanisms.

The concept of buffer-less deflection routers for NoCs is first proposed by Nilsson et al. [11] and later extended in BLESS [8]. BLESS uses a sequential port prioritization mechanism that increases the critical path delay of the router pipeline. CHIPPER [1] employs a parallel port allocation scheme with a better pipeline stage delay at the expense of increased deflection rate. But the golden packet scheme used for prioritizing packets is not effective.

Automatic Flow control (AFC) [12] is a hybrid approach that uses a conventional buffered router with a provision to switch to buffer-less mode under low network load by using the power gating technique. The Flexi-buffer design [13] uses fine grained power gating thereby adjusting the size of the active buffers adaptively. Both these techniques achieve dynamic power reduction, but the router area remains same due to the physical presence of buffers.

The central and the ring deflection algorithms proposed in [14] use sequential port allocation techniques, which increase the router critical paths. The ring algorithm deflects flits away from the center of the mesh thereby reducing the formation of hotspots. MinBD [7] is a promising solution that effectively combines the merits of buffer-less and buffered routing mechanisms. Rather than using buffers in input channel, MinBD employs a minimal set of side buffers that can accommodate one among the deflected flits per cycle.

An exhaustive study on the congestion issues in buffer-less NoCs with hundreds of workloads and application mixes is done in [15]. A detailed comparison of various design parameters of the buffered and the buffer-less paradigms is covered in [16]. Our experimental studies on various real and synthetic workloads show that the metric that decides when and whether to buffer or deflect a flit has significant impact on the average flit latency, deflection rate, and overall performance. We try to address this issue by introducing a novel hybrid deflection router, DeBAR.

III. Motivation

The existing best baseline deflection router, MinBD, is proposed to address the limitations of BLESS and CHIPPER. To reduce the deflection rate of these basic deflection routers at high network loads, MinBD technique uses the side buffer concept to store one of the deflected flits per cycle. Even though MinBD design claims a promising improvement in performance, many unoptimized design parameters expose the inability of MinBD in attaining its real potential.

The golden packet concept in CHIPPER used for prioritizing the flits is not effective because more than 95% of flits are delivered without becoming golden. Hence the golden priority status is not impacting the common case performance. Majority of the routers are dealing with non-golden flits only at any given time. Even though MinBD design proposes a silver flit concept along with the golden packet to tackle this issue, we find that the silver flit method is also inefficient. A silver flit gets the desired port in a given router. Since the silver flit is randomly selected and the silver status is strictly local (not propagated to other routers), there is no guarantee that the flit gets a silver status in the next router. This could lead to a situation where a flit gets a productive port (moving closer to the destination) when it is a silver flit in router R and deflected away from the destination if it turns out to be a non-silver flit in the adjacent router of R, thereby violating the ordering and progress of flits in the system.

Experimental results in [7] indicate that for 8% of cases, there are at least two flits destined to a local router in the same cycle. MinBD keeps dual ejection ports for the parallel ejection of these two flits rather than deflecting one of them by employing a single ejection port. This keeps one of the ejection ports idle for 92% of the time, which is quite unfair.

In MinBD design, the flit injection from core buffer into router pipeline can happen only if at least one of the input flit channels is free after the side buffer re-injection stage. The router design structure gives priority to re-injections from the side buffer over injections from the core buffer. This may lead to starvation and early self-throttling of core at higher injection rates, thereby degrading the respective application's performance. The side buffer purging [7] technique takes care of starvation issues of the side buffer alone, leaving the core buffer starvation unattended.

Resolving arbitration by random selection of flits at various stages of the router pipeline in MinBD and CHIPPER affects the network latency badly. The random selection of flits is done at buffer purging, silver flits selection, and selection of flits for side buffering. The justification given by the authors for random selection is the circuit simplicity. But a simple priority based flit selection strategy (to be discussed in Section IV-D) can be used without affecting the critical path delay.

MinBD uses a 4-flit side buffer on all the routers in the mesh network. But considering the general traffic load, the corner and edge routers carry less traffic than the central ones. So we argue that allocating unequal number of side buffers across the routers based on their physical position within a chip can yield better power-performance results.

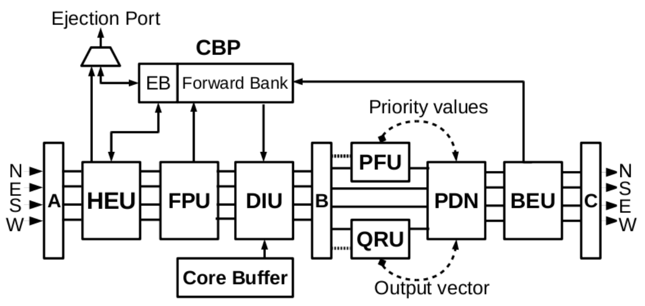

Fig. 1: Router pipeline for DeBAR. HEU-Hybrid Ejection Unit, FPU-Flit Preemption Unit, DIU-Dual Injection Unit, PFU-Priority Fixer Unit, QRU-Quadrant Routing Unit, PDN-Permutation Deflection Network, BEU-Buffer Ejection Unit, CBP-Central Buffer Pool. A, B, and C are pipeline registers.

IV. DeBAR Architecture

DeBAR is a 2-stage deflection router which uses a minimal central pool of buffers to accommodate a fraction of mis-routed flits. A block diagram of various stages of the router pipeline of DeBAR is shown in Figure 1. Detailed working of various units is discussed in the following sections. The four internal flit channels carry the input flits through various units of the router pipeline. At the end of each clock cycle, the flits are stored in the corresponding pipeline registers.

A. *Hybrid Ejection Unit (HEU)*

At the beginning of each cycle, flits from various neighboring routers reach the input pipeline register A. HEU identifies the flits intended to the local core. When there is a single ejection flit in the current cycle, the flit is removed from the internal flit channel and is forwarded to the ejection port. If the Ejection Bank (EB) in the Central Buffer Pool (CBP) is empty, HEU can handle at most two flit ejections in the same cycle. One of the flits to be ejected moves out to the local ejection port and the other to EB. The flit that is sent to EB moves to the ejection port in the subsequent cycle without any further

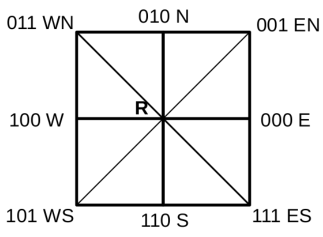

Fig. 2: Output-vector generation for quadrant routing.

TABLE I: Marking bit generation logic based on allocated output port in PDN and output-vector. 0 indicates productive port and 1 indicates non-productive port.

Port Allocation in PDN	**Output Vector**							
	000 E	**001 EN**	**010 N**	**011 WN**	**100 W**	**101 WS**	**110 S**	**111 WS**
East (000)	0	0	1	1	1	1	1	0
North (010)	1	0	0	0	1	1	1	1
West (100)	1	1	1	0	0	0	1	1
South (110)	1	1	1	1	1	0	0	0

delay. If EB already has a flit (a locally destined flit that is stored in the previous cycle), at most one flit could be moved out of the internal flit channel for ejection. In this case the flit from EB moves to the ejection port and the flit selected for ejection will move to EB. Thus HEU is capable of ejecting two flits with a single ejection port. This dual ejection mechanism fails only if there are multiple flits destined for the local core for consecutive cycles (which is a very rare possibility). In such case HEU ejects only one flit for that cycle.

B. *Dual Injection Unit (DIU)*

Flits are injected to the router pipeline from two locations- (1) core buffer (2) forward bank of CBP. Flits generated from the local core are kept in the core buffer until the router pipeline has a free slot for injection. The forward bank of CBP holds the misrouted and preempted flits in previous cycles that are to be re-injected. DeBAR handles flits from both these buffers with equal priority so that flits from neither of them will starve out to get into the pipeline.

DIU injects a flit into the pipeline if at least one of the internal flit channels is idle after passing through HEU. If there are at least two idle slots in the internal flit channel, flits are injected to the router from both these buffers, thereby performing dual injection. But if there is only one idle slot, preference is given for injection from the forward bank of CBP in odd cycles and that from the core buffer in even cycles. This odd-even cycle based priority ensures balanced progress of flits from both the buffers. DIU completes the stage 1 of the router pipeline and the flits reach the pipeline register B.

C. *Flit Preemption Unit (FPU)*

At higher injection rates, a situation can arise for few consecutive clock cycles where a router gets flits from all its neighbors. This makes all input ports busy thereby preventing injections by DIU which in turn saturates CBP and the core buffer. Core buffer saturation leads to application throttling and CBP saturation prevents further buffering of deflected flits. This not only increases the deflection rate but also blocks flits presently residing in CBP. This violates the ordering rule as the older flits are trapped in CBP and the newer flits make progress through deflection. To avoid such situations, our FPU takes necessary remedial steps.

We define the time since the last injection from the CBP as *Re-Inject Interval* (RII) and that from the core buffer as *Core Inject Interval* (CII). We also define an upper threshold to both RII and CII such that when at least one of them reaches the threshold, FPU preempts one flit from the internal flit channel and places it in the forward bank of CBP. This preempted flit will eventually return to the router pipeline as per the re-injection policy used. By this preemption, an empty slot is created in the internal flit channel so that injection or re-injection from the respective buffer can be resumed, thereby avoiding starvation. This threshold based preemption prevents the overflow of the CBP and the core buffer. The choice of which among the incoming flits is to be preempted is chosen by the preemption metric. We fix both CII and RII to 2 cycles.

D. *Priority Fixer Unit (PFU)*

The role of PFU is to assign a total ordering to all the flits residing in the pipeline register B. By this total ordering we ensure fairness and progress in the flit movement. PFU extracts the destination address field from all the flits residing in register B and computes the priority value. Flits closer to the destination are given highest priority. PFU assigns one of the three different priority levels: 0 - for the flits whose destination is within 2-hop distance; 1 - for the flits whose destination is between 2- and 4-hop distance; and 2 - for the flits whose destination is more than 4-hops away. A flit with the highest priority is handled similar to a silver flit in MinBD router.

E. *Quadrant Routing Unit (QRU)*

Similar to PFU, QRU also extracts the destination address field from all the flits residing in the register B. Based on the destination address of the flit, a 3-bit output-vector is computed. This output-vector indicates the possible productive ports for the flit. Figure 2 shows the logic behind the generation of the output-vector. As per the figure, let R be the position of the current router. Center of the edges (E, N, W, and S) and corners (EN, WN, WS, and ES) represent the relative position of the destination (D). If R and D belong to the same row or column, the flit at R has exactly one productive port and the flit is assigned an output-vector which is an even number (000, 010, 100, or 110). On the other hand, if they both have different row and column, the flit can have two productive ports and the output-vector assigned is an odd number (001, 011, 101, or 111). The computed priority value from PFU and the output-vector value from QRU are forwarded to Permutation Deflection Network (PDN). The generated output-vector for a flit is used only within the current router and once the flit leaves register C, the values are reset.

F. Permutation Deflection Network (PDN)

Similar to the one used in CHIPPER and MinBD for the parallel allocation of output ports, DeBAR also uses PDN, a two stage arbitration circuit. But the PDN in DeBAR has two additional control logic units in it: (1) the header enhancer circuit that adds the priority value and the output-vector to the flit header and (2) the flit marking circuit that identifies the mis-routed flits from others. For each arbitration stage of PDN, the priority levels and the output-vectors of the incoming flits decide the port allocation of that arbiter stage.

The highest priority flit always gets the productive port. By the destination-hop based priority scheme, DeBAR makes sure that the flits that are near to the destination are not deflected. Sometimes, depending on the contention level and port conflicts, other flits may not get a productive port. The flit marking circuit identifies whether the port allocation done by PDN leads to mis-routing or not. It takes the allocated output port and the output-vector as inputs and generates the value of the marking bit as shown in Table I.

G. Buffer Ejection Unit (BEU)

Flits marked with 1 indicate that they are assigned non-productive ports which take them away from the destination. From among the flits coming out from PDN, BEU selects at most one flit marked with 1 for storing into the forward bank of CBP. By this central buffering, we reduce the average deflection rate of the network thereby bringing down unwanted flit movements in the network. Flits that are not buffered come out through BEU to the respective output links based on the allocation done by PDN. Once a flit leaves a router to its neighbor this marking is cleared.

H. Core Buffer & Central Buffer Pool (CBP)

DeBAR contains two sets of buffers: core buffer and CBP. Deflection routers work with a specific injection policy that the newly generated flits from a local core can be injected into the router only if there is a free slot in the internal flit channel. As long as free slots are not available, the generated flits from the local core will be kept in the core buffer. DIU takes care of flit injection from the core buffer.

DeBAR uses the CBP to store the deflected and preempted flits. This is similar to the side buffer concept used in MinBD [7]. We propose a non-uniform count of buffers in the CBP across various routers in mesh network based on the physical location of the router. Size of the CBP for central, edge and corner routers in a mesh NoC are 4, 3, and 2, respectively. CBP is partitioned into two banks, ejection bank and forward bank. As long as multiple ejection flits are not there in a cycle, the entire space in CBP is allocated to the forward bank. When multiple flits are to be ejected in the same cycle, one buffer from CBP is assigned to form an ejection bank. Flits in the ejection bank of CBP move to ejection port in the subsequent cycle and flits in the forward bank get re-injected to the router pipeline as per re-injection policy.

TABLE II: Percentage of different network injection intensity applications in various benchmark mixes.

Benchmark Mix	**M1**	**M2**	**M3**	**M4**	**M5**	**M6**	**M7**
% of Low	100	0	0	50	0	50	31
% of Medium	0	100	0	0	50	50	31
% of High	0	0	100	50	50	0	38

V. Experimental Methodology

We modify the traditional VC based NoC simulator *Booksim* [3] to model the two-cycle deflection router micro-architecture mentioned in CHIPPER [1] with sufficient detail. We consider flits with header information attached to it. This facilitates independent routing of flits within a same packet as it is the common standard in deflection routers. We use necessary reassembly mechanism for handling out-of-order delivery of flits. The flit channel is 140-bit wide: 128-bit data field and 12-bit header field. On this baseline deflection router simulator we make changes to model MinBD router design as in [7] and DeBAR design and conduct experimental analysis.

A. Real Workloads

We use Multi2sim [17] simulator to model a 64-core CMP setup with CPU cores, cache hierarchy, and coherence protocols in sufficient detail and accuracy. Each core consists of an out-of-order x86 processing unit with a 64KB, 4-way set-associative, 32 byte block, private L1 cache and a 512KB, 16-way set associative, 64 byte block, shared distributed L2 cache. Each core is assigned with a SPEC CPU2006 benchmark application for running on it. Based on the *misses per kilo instructions* (MPKI) values calculated on a 64KB L1 cache, we classify the benchmarks into *Low* (MPKI less than 5), *Medium* (MPKI between 5 and 25) and *High* (MPKI greater than 25). We construct 35 multiprogrammed workloads, each with 64 single threaded benchmark instances. We categorize these workloads into 7 mixes (M1 to M7) based on the proportion of the network injection intensity *(Low / Medium / High)* of the constituent benchmarks. Details of the mixes are given in Table II. After sufficient fast forwarding, we capture the L1 cache misses that generate network traffic and feed it to the modified Booksim model to simulate the network operations.

B. Synthetic Workloads

In order to show the robustness of DeBAR, we also evaluate our design using seven standard synthetic traffic patterns: *uniform*, *transpose*, *tornado*, *bit-complement*, *bit-reverse*, *shuffle*, and *neighbor* for 8x8 mesh network. Average packet latency, deflection rate, and throughput values are collected for each traffic pattern with injection rate varying from zero to saturation. We plot results only for few traffic patterns due to space constraints.

VI. Experimental Analysis

We compare the performance of DeBAR with MinBD router and a traditional input buffered VC Router (VCR). We consider 16 VCs per input port and uses XY routing for VCR.

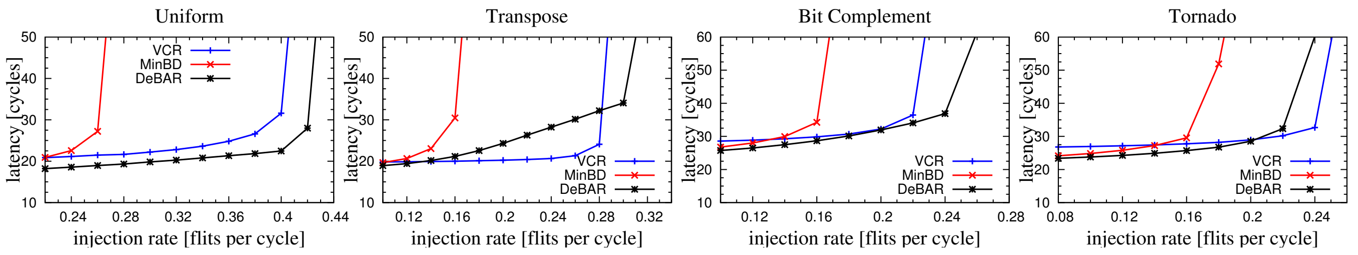

Fig. 3: Average flit latency comparison under various synthetic traffic patterns in 8x8 network.

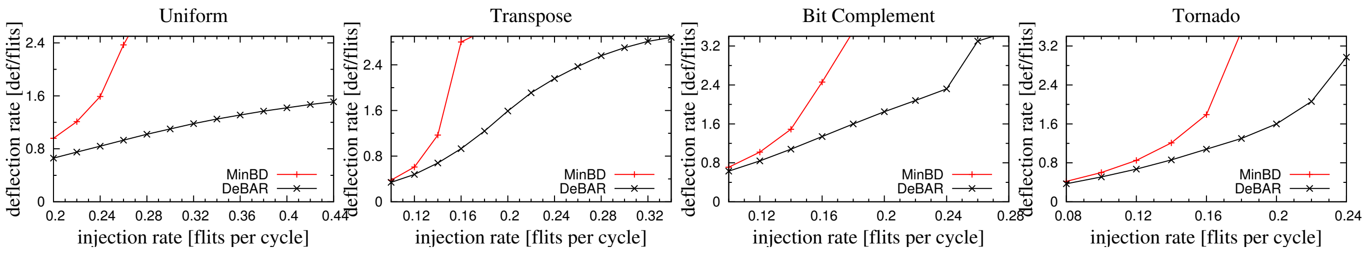

Fig. 4: Deflection-rate comparison under various synthetic traffic patterns in 8x8 network.

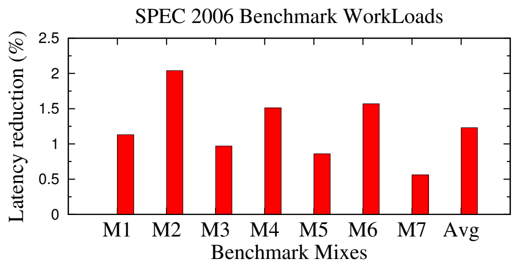

Fig. 5: Percentage reduction in average flit latency w.r.t. MinBD.

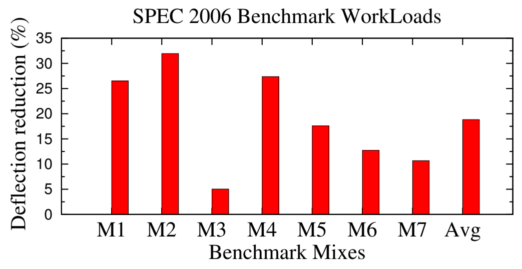

Fig. 6: Percentage reduction in deflection-rate w.r.t. MinBD.

Comparison with the VCR technique helps us to understand how closely a minimally buffered adaptive deflection router like DeBAR can perform with an input buffered router with static routing. For MinBD, we consider a 4-flit side buffer.

A. Effect on Average Flit Latency

Figure 3 shows a set of injection rate- flit latency graphs with MinBD, DeBAR and VCR on an 8x8 mesh network using the synthetic traffic patterns. From the plots we can see that, across all these patterns DeBAR shows lower average flit latency than MinBD and it is very close to the that of VCR. The static XY routing of VCR makes it more congestion prone and hence it tends to saturate early whereas DeBAR by virtue of the deflection mechanism spreads traffic evenly. In three out of four patterns, DeBAR extends the saturation injection than VCR. This makes DeBAR a good design choice for high injection rate applications.

Figure 5 shows the percentage reduction in average flit latency with respect to MinBD for various application workloads specified in Section V-A. For all mixes, we can see the reduction in the average flit latency using DeBAR design. The latency reduction is less in mixes M3, M5, and M7 as they consist of high MPKI applications like *hmmer, lbm, leslie3d,* and *mcf*, which contribute more network traffic than other mixes with low and medium MPKI applications (*calculix, gromacs, bwaves, gcc, h264ref*).

B. Effect on Deflection Rate and Throughput

Deflection rate is defined as the average number of deflections per flit. As injection rate increases, deflection rate also increases due to high port contention. From Figure 4, we can observe that, DeBAR achieves less deflection rate as compared to MinBD for all synthetic traffic patterns. This is due to the priority scheme adopted in our scheme that prevents the deflection of flits once they are near to the destination. The central buffering based on marking chooses the right candidate for buffering and leaves out others for mis-routing. This also contributes to the low deflection rate in DeBAR.

In deflection rate analysis of the real workloads (refer to Figure 6), we can see that DeBAR outperforms MinBD. In workloads with low MPKI applications (M1, M2, and M4), the reduction in deflection rate is around 30%. This reduces the network activity factor and helps in dynamic power savings.

We compute the throughput as the number of flits ejected per router per cycle. Figure 7 contains the throughput plots. Due to early network saturation and high deflection rate,

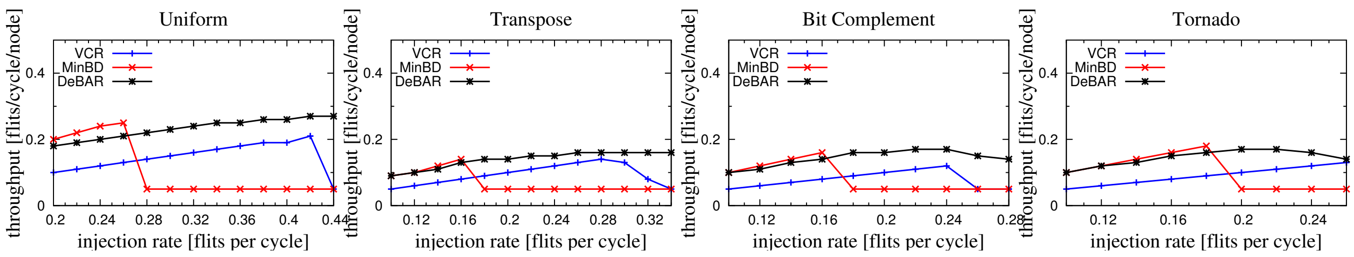

Fig. 7: Throughput comparison under various synthetic traffic patterns in 8x8 network.

MinBD experiences a sharp dip in the throughput. Similarly the VCR also encounter a dip in throughput when the network saturates. With DeBAR design, flits will reach destination without much deflection due to the unique priority scheme and the selective central buffering of the mis-routed flits. By deflection routing, DeBAR exploits non-minimal paths that helps it to maintain a better throughput than VCR.

C. *Effect on Router Critical Path, Area, and Power*

We implement the pipelined design of CHIPPER, MinBD, and DeBAR in Verilog and synthesize using Synopsys Design Compiler with 65nm library to obtain the timing latency for each unit in the pipeline stage. HEU is taking 34% less timing latency than the two levels of ejection units in MinBD. The DIU unit is having 11.5% increase in timing latency than the combined buffer inject and normal inject units of MinBD. Overall, in stage 1 of the router pipeline, DeBAR reduces timing latency by 17% than MinBD.

In stage 2, out of the two parallel units, PFU dominates the critical path latency. It takes 38% more time than the random silver block in MinBD. This additional latency is compensated by QRU logic that works in parallel to PFU. QRU takes away the route finding logic of the conventional PDN used in MinBD and CHIPPER. Even though our PDN uses the extra logic for header enhancement and marking, removal of route finding logic (QRU) from the critical path save 30% latency in PDN circuit. Our timing analysis finds that overall latency of stage 2 is same for MinBD and DeBAR. The latency in stage 2 dominates that of stage 1 in both MinBD and DeBAR. Hence we argue that DeBAR can be operated at the same network frequency as that of MinBD.

We compute the area and power estimates of DeBAR using Orion 2.0 [18]. We assume 65nm technology at 1GHz operating frequency with an NoC channel delay of one cycle [19]. Power dissipation and router area in DeBAR are same as that of MinBD since the buffering and control logic complexity within a single router are same for both the designs. The non-uniform CBP count across routers brings 5% power savings with respect to MinBD in the total network. Due to the absence of dual ejection port our design reduces channel wiring overhead by 18% with respect to MinBD.

VII. Conclusion

We proposed a novel deflection router with minimal central buffering. Performance limitations in existing baseline models are rectified with a superior design that dynamically decides whether to buffer/deflect/assign a productive port to an incoming flit. Injection of newly generated flits and re-injection of buffered and preempted flits are coordinated in an effective manner with better priority metrics. Experimental results showed that our design not only reduces the average flit latency, but also reduces the deflection rate, and increases throughput. All these design optimizations make DeBAR an excellent choice for minimally buffered NoC routers.

VIII. Acknowledgment

This work is supported in part by grant from Defence Research and Development Organization (DRDO) under IITM-DRDO MoC.

References

[1] C. Fallin et al., "CHIPPER: A low complexity bufferless deflection router," in *HPCA*, 2011, pp. 144–155.

[2] W. Dally, "Virtual-channel flow control," *IEEE Transactions on Parallel and Distributed Systems*, vol. 3, no. 2, pp. 194–205, 1992.

[3] W. Dally and B. Towles, *Principles and Practices of Interconnection Networks*. USA: Morgan Kaufmann Publishers Inc., 2003.

[4] Y. Hoskote et al., "A 5-GHz mesh interconnect for a teraflops processor," *IEEE Micro*, vol. 27, no. 5, pp. 51–61, 2007.

[5] M. B. Taylor et al., "Evaluation of the raw microprocessor: An exposed-wire-delay architecture for ILP and streams," in *ISCA*, 2004.

[6] "SPEC2006 CPU benchmark suite," http://www.spec.org.

[7] C. Fallin et al., "MinBD: Minimally-buffered deflection routing for energy-efficient interconnect," in *NOCS*, 2012, pp. 1–10.

[8] T. Moscibroda and O. Mutlu, "A case for bufferless routing in on-chip networks," in *ISCA*, 2009, pp. 196–207.

[9] M. Hayenga et al., "SCARAB: A single cycle adaptive routing and bufferless network," in *MICRO*, 2009, pp. 244–254.

[10] C. Gomez et al., "An efficient switching technique for NoCs with reduced buffer requirements," in *ICPADS*, 2008, pp. 713–720.

[11] E. Nilsson et al., "Load distribution with the proximity congestion awareness in a network-on-chip," in *DATE*, 2003, pp. 1126–1127.

[12] S. A. R. Jafri et al., "Adaptive flow control for robust performance and energy," in *MICRO*, 2010, pp. 433–444.

[13] G. Kim et al., "Flexibuffer : Reducing leakage power in on-chip network routers," in *DAC*, 2011, pp. 936–941.

[14] G. Oxman et al., "Streamlined network-on-chip for multicore embedded architectures," in *ARCS*, 2012, pp. 238–249.

[15] G. Nychis et al., "Next generation on-chip networks : What kind of congestion control do we need?" in *Hotnets'10*, 2010, pp. 1–6.

[16] Z. Lu et al., "Evaluation of on-chip networks using deflection routing," in *GLSVLSI'06*, 2006, pp. 296–301.

[17] R. Ubal et al., "Multi2sim: A simulation framework to evaluate multicore-multithreaded processors," in *SBAC-PAD*, 2007, pp. 62–68.

[18] A. B. Kahng et al., "Orion 2.0: A fast and accurate NoC power and area model for early stage design space exploration." in *DATE*, 2009, pp. 423–429.

[19] W. Zhao and Y. Cao, "Predictive technology model for nano-CMOS design exploration," *ACM Journal on Emerging Technologies in Computing Systems*, vol. 3, pp. 1–17, 2007.