Write Variation Aware Non-Volatile Buffers for On-Chip Interconnects

Khushboo Rani and Hemangee K. Kapoor
Department of Computer Science and Engineering, IIT Guwahati, Assam, India-781039
Email: {khushboo, hemangee}@iitg.ac.in

***Abstract*—With the advancement in CMOS technology and multiple processors on the chip, communication across these cores is managed by a Network on Chip (NoC). Power and performance of these NoC interconnect have become a significant factor. In particular, the buffers used at every port consume considerable dynamic as well as leakage power.**

This paper attempts to reduce leakage power consumption of NoC buffers by use of non-volatile memory (NVM) technology based STT-RAM buffers. STT-RAM technology has the advantage of high density and low leakage but suffers from high write energy and low endurance. In particular, if some buffers get written more number of times compared to some other buffers, the heavily written buffers may wear out faster compared to others. This has an impact on the lifetime of the router as a whole. Here we propose a virtual channel (VC) allocation policy that helps to uniformly distribute the writes across the buffers to improve their effective lifetime. Pure STT-RAM buffers, however, impact the network latency. In order to mitigate this, we propose a hybrid policy that uses an alternative VC made of SRAM technology in case of heavy network traffic. Experimental evaluation on full system simulation shows that the proposed policy reduces the write variation by 99% and improves lifetime by 3.4 times and 24 times, respectively for both the proposals. We also get respectively 93% and 90% gains in the EDP.

***Index Terms*—Network-on-Chip, STT-RAM, NVM, buffer, leakage power, write variation, allocation policy**

I. Introduction

With the continuing advancement in CMOS technology, we have more and more transistors packed on the same die leading to multiple processing cores on the chip known as chip multiprocessors (CMPs). Communication across multiple cores happens with the help of switch-based Network on Chip (NoC). Design space of NoC is tightly constrained as the chip real estate needs to be distributed across the multiple cores as well as multiple levels of caches. The same constraint applies to power consumption. It has been noticed that communication consumes upto 39% [1] of total chip power. With tighter power budgets and to meet the thermal design power (TDP) for the system, components like the cores/caches undergo voltage and frequency scaling and at times power off. Powering off several components to stay within the TDP leads to the concept of dark silicon. In dark silicon, although the cores/caches are off, the communication network is expected to be available. In order to reduce the standby power of the network in such events, one looks for avenues in non-volatile memory (NVM) technologies. Attempts have been made in using a combination of NVM and SRAM buffers in the routers [2] [3] for energy efficiency. The challenge in this type of work is the effective utilization of the buffers. NVM technologies such as Resistive Random Access Memory (ReRAM), Spin Transfer Torque Random Access Memory (STT-RAM) and Phase Change Random Access Memory (PRAM) offer many advantages over the conventional SRAM technology. These advantages include high density, good scalability, and low leakage power consumption. However, the buffers made from these memory technologies suffer from costly write operation and low write endurance. The write endurance values of these NVMs are 10^8 writes for PRAM, 10^{11} writes for ReRAM and, for STT-RAM the predicted value is 10^{15} writes but the write endurance value tested so far is 4×10^{12} writes [4] [5] [6]. Further, the weak endurance of the NVM buffers can also get affected by the unwanted write variations generated by the system. In other words, if we design NoC buffers using STT-RAM then the number of writes to them will be governed by the NoC policies and this variation will affect the lifetime of the individual buffers: as some buffers may get written several times more compared to some other buffers. This indirectly will affect the lifetime of the router as a whole. Towards solving this issue, this paper proposes a write variation-aware virtual channel (VC) allocation policy for NoC routers. A baseline NoC router has buffers at its input and output ports. The physical links are logically partitioned into virtual networks (VNets) which in turn are divided into virtual channels (VCs). The different VNets considered in this paper represent *Control, Response* and *Data*. In MESI protocol the cache coherence packets are categorized into control and response and these are transmitted on separate VNets. The buffers inside a VC of the VNet store the incoming flits and depending on the VC allocation policy the number of writes incurred by the buffers may vary. We quantify this by defining the coefficient of write variation in section III-B. Using this definition and running experiments on a number of benchmark programs we obtain results as shown in figure 1. It can be seen that the write variation is quite significant across VCs belonging to the same VNet. This

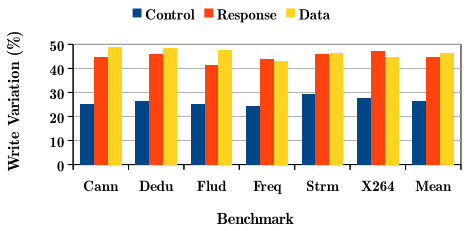


Fig. 1: Write variation in STT-RAM baseline with 64 core, 8×8 mesh network for different virtual networks.

2380-6923/19/$31.00 ©2019 IEEE
DOI 10.1109/VLSID.2019.00020

variation can be controlled by evenly distributing the writes across the VCs by using an appropriate VC allocation policy. The major contributions of this paper are:

- We present a VC allocation policy, WVAR, to minimize the write variation across the buffers in the VCs.
- In order to mitigate packet latency degradation due to use of long latency NVM buffers, we propose a hybrid policy, Hy-WVAR, that uses an alternate SRAM VC to be used in case of heavy traffic.
- Full system simulation results using Gem5 [7] and Garnet [8] as the NoC simulator shows 3.2 and 22 times improvement in the lifetime for WVAR and Hy-WVAR respectively and 99% reduction in write variation compared to the baseline.

II. Background

A. STT-RAM

Figure 2 shows the representational view of STT-RAM [9] cell. The cell consists of magnetic storage element called as Magnetic Tunnel Junction (MTJ) series with access transistor. MTJ accommodates tunnel barrier (usually MgO) two ferromagnetic layers (free and reference layers). The tunnel barrier is in center of the two ferromagnetic layers. The magnetization direction of the reference layer is fixed while the direction of the free layer is changed according to spin-polarized current. The difference in the direction of two ferromagnetic layers is used to represent the data bit stored in the MTJ. The parallel magnetization direction of ferromagnetic layers represents low resistance and corresponds to '0' state of STT-RAM cell. On the other hand, anti-parallel magnetization direction represents high resistance and corresponds to '1' state. Details about read/write operation and latency can be referred from [9].

B. Related Work

NoC buffer power and area optimization: Input buffers consume upto 45% of router power and occupy 15% of router area [10]. In buffered routers, increasing buffer size increases leakage power and area overhead while reducing buffer size causes performance degradation. In order to reduce buffer power, bufferless [11] and elastic-buffered NoC [12] have been proposed. The idea is to remove buffers from the router to reduce leakage power. Bufferless techniques are not scalable as they suffer from performance degradation when load increases. Bufferless approach also does not support VCs. This makes impossible to implement multiple traffic classes which are needed to avoid protocol-level deadlock. Instead of removing the buffer from the router [13] decreased buffer size and repeaters were employed to work as a buffer during network congestion. The idea in [14] proposed to replace conventional SRAM buffers with eDRAM. This reduced power by 43% and area by 52%.

STT-RAM as input buffer of on-chip routers : STT-RAM based caches [15] and hybrid caches [16] [17]idea is well explored. There are few researches done on leveraging STT-RAM in on-chip interconnects which reduces leakage power and area of the router. In [3] authors propose large STT-RAM based NoC buffer to enhance performance. The idea is lazy migration, to hide write latency of STT-RAM in the hybrid design of input buffers combining SRAM and STT-RAM. Due to high write, energy consumption of STT-RAM writes, migrating each flit from SRAM to STT-RAM is not feasible. Lazy migration allows flit migration when network load exceeds the threshold. This technique reduces power by 61% on average compared to simple migration of flits from SRAM to STT-RAM. This method suffers from power overhead in high network load. In [2] authors have replaced conventional SRAM input buffer on-chip with a combination of drowsy SRAM and STT-RAM buffer. They have proposed two hybrid drowsy SRAM / STT-RAM buffer designs, hybrid buffer and banked buffer to reduce write latency of STT-RAM. Utilizing advantages of both drowsy SRAM and STT-RAM.

None of the above methods address the issue of the number of writes and lifetime of STT-RAM buffers. Our paper addresses this by proposing the VC allocation scheme.

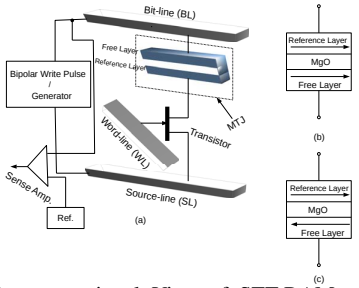


Fig. 2: (a) Representational View of STT-RAM cell (b) Parallel low resistance, representing '0' state (c) Anti-parallel high resistance, representing '1' state.

III. Proposed Technique

This section presents the baseline VC allocation policy and the proposed allocation policies: *WVAR* and *Hy-WVAR*. We also define the coefficient of write variation.

A. Motivation and Basic VC Allocation

Here we discuss the process adopted by general purpose NoC routers to allocate VCs at the ports for the incoming flits. After route computation, the flit goes through the Virtual Channel (VC) allocation stage. Each port uses wormhole routing and has multiple VCs per input port. Credit-based flow control is used to avoid buffer overflow and also guarantees free buffer space at a downstream input port. This also controls flit transmission and buffer overflow as well as packet loss. The buffers in NoC are very simple and mostly FIFO based having, say, k number of entries. These buffers are managed using head and tail pointers so that the flits do not need to traverse all the buffers before exiting. VCs are logically grouped under a virtual network (VNet). Packet/flit travels from one router to another over one virtual network to avoid protocol-level deadlock. VC allocator assigns the packet to a VC belonging to the appropriate class (i.e. VNet), similar to the one of the downstream router. Within the VNet the VC allocator looks for a free/available VC to be assigned to the packet. As the flits are output, the VCs become available to be used by the subsequent packets. Thus, it may happen that a particular VC

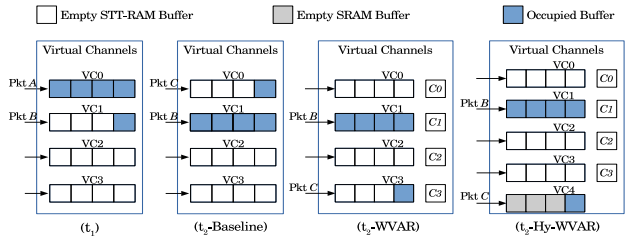


Fig. 3: Virtual network showing VC allocation at different time stamp in baseline and proposed architectures.

within a VNet incurs more writes compared to other VCs in the same VNet.

Figure 3 shows an example scenario. At time-stamp t_1, first packet, pkt_A, gets allocated to virtual channel vc_0. When second packet, pkt_B, arrives it gets allocated to vc_1. By the time third packet arrives, i.e. at time-stamp t_2, vc_0 becomes free and hence VC allocator allocates vc_0 to the packet pkt_C (as shown in part *(t_2-Baseline)* of the figure); whereas channels vc_2 and vc_3 are idle. In this approach, VC allocator chooses the first free VC to allocate packets, thus increasing the number of writes on a particular virtual channel (vc_0 in this case).

This allocation policy is fine as long as we have long endurance SRAM buffers. However, with NVM buffers one needs to consider the buffer's lifetime. Note that buffer's lifetime is the inverse of the maximum number of writes on that buffer. In particular, the heavily written buffers wear out faster compared to other buffers, leading to a reduction in the overall lifetime. This, in turn, affects the lifetime of the VC and the router.

B. Write Variation

In order to quantify the disparity in the number of writes, in this section, we formally define write-variation. Let N be the number of routers, I number of input ports per router, V number of a virtual network at each input port and A number of virtual channel in each virtual network. Also, W_j denotes the average number of writes in all virtual channel in virtual network j and w_{ji} denotes the number of writes in virtual channel i in VNet j.

Similar concept is defined for caches designed using NVMs [18]. We adapt the definition and define the coefficient of write variation across the buffers in a given VNet, j, as follows:

$$Var_j = \frac{100}{W_j}\sqrt{\sum_{i=1}^{A}\frac{\left(w_{ji} - \sum_{k=1}^{A} w_{jk}/A\right)^2}{A-1}} \quad (1)$$

The average variation for each virtual network across all routers for each input port will be defined as:

$$Avg_j = \frac{\sum_{n=1}^{N}\sum_{i=1}^{I} Var_j}{N * I} \quad (2)$$

C. Proposed Technique-1: WVAR

The main aim of our proposed technique is to reduce the write variation across the buffers inside a VNet. In order to uniformly distribute the writes, we maintain counters w_{va} with each input port, virtual network v and virtual channel a. This count is incremented each time a write happens to a buffer in VC a. If we have these write counts for all VCs within a VNet, we can uniformly distribute the writes. In particular, during the VC allocation stage, among the free VCs, the VC with minimum write counts is allocated to the packet. The detailed algorithm is given in Algorithm 1 and is called as Write VARiation aware VC allocation (WVAR).

Algorithm 1 WVAR Algorithm

1: I : Number of Input ports
2: V : Number of Virtual networks at input port
3: A_v : Set of Virtual channels at virtual network v
4: w_{va} : Number of writes in virtual channel a of virtual network v
5: $w_{va} = w_{va} + 1$, when a new flit is buffered in VC a of virtual network v
6: Store w_{va} for each VC a of virtual network v for every downstream router
7: **for** Every VC allocation stage **do**
8: Find free/idle VCs ($A_v^{'}$) in virtual network v, where $A_v^{'} \subset A_v$
9: **for** $\forall a_i \mid a_i \in A_v^{'}$ **do**
10: Find $a_j \mid w_{va_j} = min(w_{va_p}), \forall a_p \in A_v^{'}$
11: **if** $\exists\ a_j$ and $a_k \mid w_{va_j} = w_{va_k} = min(w_{va_p}), \forall a_p \in A_v^{'}$ and $j \neq k$ **then**
12: Select VC in Round Robin manner among a_j, a_k
13: **end if**
14: Allocate VC
15: **end for**
16: **end for**

Algorithm 2 Hy-WVAR Algorithm

1: I : Number of Input ports
2: V : Number of Virtual networks at input port
3: Int : Number of cycles defining an interval of execution
4: A_v : Set of Virtual channels at virtual network v
5: A^{stt} : Set of STT-RAM Virtual channels at virtual network made up of STT-RAM
6: A^{sram} : Set of SRAM Virtual channels at virtual network made up of RAM
7: Th_v : Threshold network traffic for virtual network
8: w_{va} : Number of writes in virtual channel a of virtual network v
9: $w_{va} = w_{va} + 1$, when a new flit is buffered in VC a of virtual network v
10: Store w_{va} for each VC a of virtual network v for every downstream router
11: At the beginning of every interval, Int
12: **if** $network\ traffic < Th_v$ **then**
13: Find free/idle VCs ($A_v^{'}$) in virtual network v, where $A_v^{'} \subset A_v^{stt}$
14: **for** $\forall a_i \mid a_i \in A_v^{'}$ **do**
15: Find $a_j \mid w_{va_j} = min(w_{va_p}), \forall a_p \in A_v^{'}$
16: **end for**
17: **else**
18: **for** $\forall a_i \mid a_i \subset A_v^{stt}$ **do**
19: Find $a_j \mid w_{va_j} = \max(w_{va_p}), \forall a_p \in A_v^{stt}$
20: **end for**
21: **if** $\exists\ a_j$ and $a_k \mid w_{va_j} = w_{va_k} = \max(w_{va_p}), \forall a_p \in A_v^{stt}$ and $j \neq k$ **then**
22: Select VC in Round Robin manner among j, k
23: **end if**
24: Find $A_v^{'} \mid A_v^{'} \subset A_v^{stt} \cup A_v^{sram} \setminus v_j$
25: **end if**
26: **if** $\exists\ a_j$ and $a_k \mid a_j$ and $a_k \in A_v^{'}$ and $j \neq k$ **then**
27: Select VC in Round Robin manner among a_j, a_k
28: **end if**
29: Allocate VC
30: Repeat from line 11 till end of execution

In the illustration, for convenience we use C_a as the count for VC a. Consider the buffer allocation at time-stamp t_1 in figure 3 when packet-A and Packet-B are allocated to vc_0 and vc_1, respectively. At time-stamp t_2 when packet-C arrives, in our proposed policy, WVAR, we check an available/free VC

Parameter	Details
core count	64, 1GHz
topology	mesh
L1 I & D cache , L2 cache	64KB, 64KB, 16MB (64B block size)
router pipeline	3 stage
packet size	control 1 flit, data 5 flits (16B flit size)
virtual network count	3 per port with 4 VC per VNet
virtual channel depth	1 buffer for control, 4 buffers for data VC
counter size	2B

TABLE I: System and interconnect configuration

Parameter	SRAM	STT-RAM
cell size(F^2)	146	45.6
read/ write latency (cycle)	1/ 1	1/ 2
read energy/bit (pJ)	0.063	0.082
write energy/bit (pJ)	0.049	0.286
leakage power (mW)	1.797	0.044

TABLE II: SRAM and STT-RAM buffer configuration

having minimum write count and allocate the packet to this VC. In this example, write count of vc_3 is the minimum among available VCs, $C_3 = min(C_0, C_2, C_3)$, and thus Packet-C is allocated to vc_3 (as shown in part *(t_2-WVAR)* of the figure).

D. Proposed Technique-2: Hy-WVAR

The earlier technique WVAR performs VC allocation while maintaining uniform writes across all the VCs within a VNet. On account of the high write latency incurred by STT-RAM based buffers, these routers tend to increase the average packet latency compared to a SRAM based NoC. It has been experimentally observed that use of pure STT-RAM buffers increases the latency by 30% (cf. Figures 7) over baseline SRAM design in a 8x8 mesh network. In an attempt to rectify this, we propose to use an alternative SRAM VC, as SRAM has lesser write latency, during high network load. In other words, when the network traffic increases beyond a given threshold[1], our proposed method allocates an SRAM VC in place of one heavily written free STT-RAM VC. Traffic analysis is done over equally divided intervals of execution, and the decision to use SRAM is taken at the beginning of every interval. This method of using a combination of SRAM and STT-RAM VCs is called the Hybrid Write VARiation aware VC allocation (Hy-WVAR). It is detailed in Algorithm 2.

In the illustration (cf. Figure 3) consider the buffer allocation at time-stamp t_1 when packets A and B are allocated to vc_0 and vc_1 respectively. At time-stamp t_2, when packet-C arrives, in the proposed policy Hy-WVAR, if the network load is beyond the threshold, we allocate the new packet to the SRAM VC, if it is free and leave the STT-RAM, vc_2, having maximum write counts as idle.

IV. Experimental evaluation

We used Gem5 [7], a multi-core full system simulator with Garnet [8], as the interconnection network model, for NoC performance Orion 2.0 [19] was used for a router power analysis at $32nm$ technology. We evaluate a 64 core system with 8x8 mesh network with a buffer depth of data VCs 4. The detailed system configuration is given in table I. We use [3][20] to get SRAM and STT-RAM latency, read-write energy and leakage power. Table II shows all the system related parameters. We evaluate our work with PARSEC Benchmark [21]. The workloads used in the paper are canneal(Cann), dedup(Dedu), fluidanimate(Flud), freqmine(Freq), streamcluster(Strm) and x264(X264).

We show results for (1) write variation (2) lifetime (3) total energy (4) packet latency and (5) EDP. Write variation is shown separately for each virtual network as they have different data buffer depths at the input ports. The results are given for the baseline policy using basic VC allocation policy (cf. section III-A) and the proposed policies: WVAR and Hy-WVAR. Improvements in the lifetime and write variation are shown with respect to baseline using STT-RAM buffers having the same number of VCs per VNet as the proposed. Reduction in static energy and comparison with EDP and network latency is done with respect to baseline using pure SRAM buffers.

A. Results and Analysis

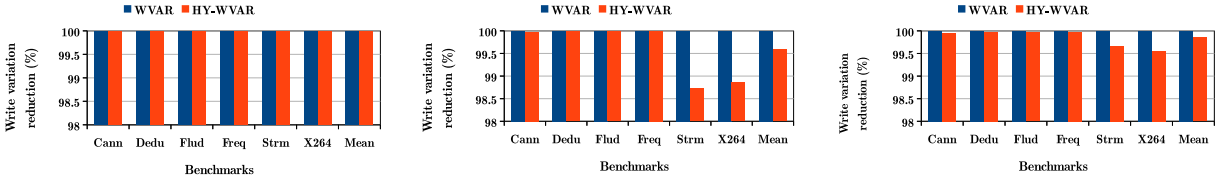
Write variation: Write variation is calculated and compared individually for each virtual networks for STT-RAM Baseline and STT-RAM Proposed methods. Figure 4a shows the reduction in write variation for the control virtual network. It is observed that the average write variation in baseline method is 26% whereas in the proposed methods it is 0.0002% in WVAR and 0.0008% in Hy-WVAR. The proposed policies thus show a reduction of almost 100% (as seen in the graph).

Write variation for response virtual network is shown in figure 4b. Even here, the gap between write variation of baseline and proposed methods is significant. Minimum write variation of STT-RAM Baseline is 41% and maximum is 47%. For Proposed methods WVAR and HY-WVAR, average write variations are 0.00008% and 0.0009% respectively. Overall average write variation for WVAR is 0.0005% and HY-WVAR is 0.02%.

For data virtual network write variation is maximum for all workloads in STT-RAM Baseline (Figure 4c). Write variation in STT-RAM Baseline are in the range of 43% to 49% and the average is 47%. Overall average write variation in the WVAR and Hy-WVAR methods are 0.0002% and 0.02% respectively. In the proposed method, write into VC for each virtual network is more evenly distributed than the default method of baseline which is evident from almost nil write variation. Uniform distribution of writes across VCs ensures the longer lifetime of the VC buffers.

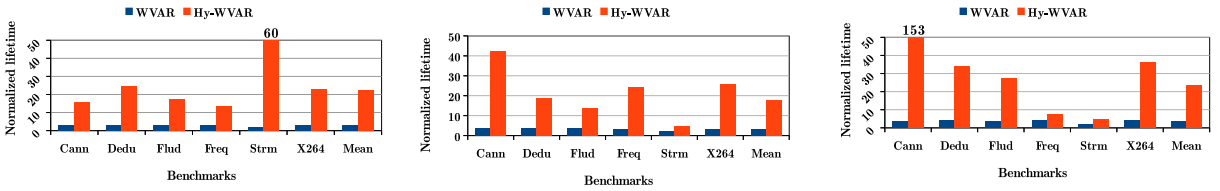
Lifetime: Similar to write variation, we have done analysis of buffer lifetime individually for each virtual network since the type of packet differs for each one. Figure 5a shows lifetime comparison of proposed method for control virtual network. In proposed WVAR technique lifetime is enhanced by 1.7 to 3.3 times. On an average lifetime is enhanced by 2.9 and 23 in WVAR and Hy-WVAR. In response virtual network, figure 5b, average life enhancement is 3.3 and 18 for WVAR and Hy-WVAR over baseline STT-RAM. For data virtual network, figure 5c, lifetime increases by factor of 3.4 and 24 on average. The use of additional SRAM buffers reduces the load on STT and hence we get more lifetime improvement in Hy-WVAR. As the data network has more

[1]Value of threshold was computed based on empirical analysis of benchmarks.



(a) Control-VNet (b) Response-VNet (c) Data-VNet

Fig. 4: Percentage reduction in write variation for proposed policies WVAR and Hy-WVAR w.r.t. STT-RAM baseline in 8×8 mesh network.



(a) Control-VNet (b) Response-VNet (c) Data-VNet

Fig. 5: Normalized lifetime improvement in STT-RAM proposed w.r.t. STT-RAM baseline in 8×8 mesh network.

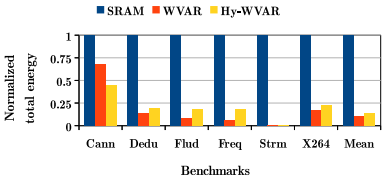


Fig. 6: Normalized total energy of proposed policies w.r.t. SRAM baseline in 8×8 mesh network.

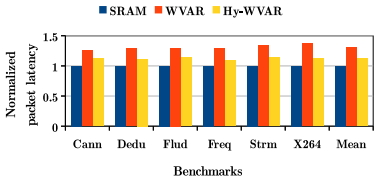


Fig. 7: Normalized packet latency of proposed policies w.r.t. SRAM baseline in 8×8 mesh network.

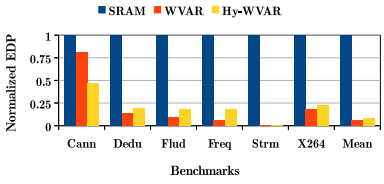
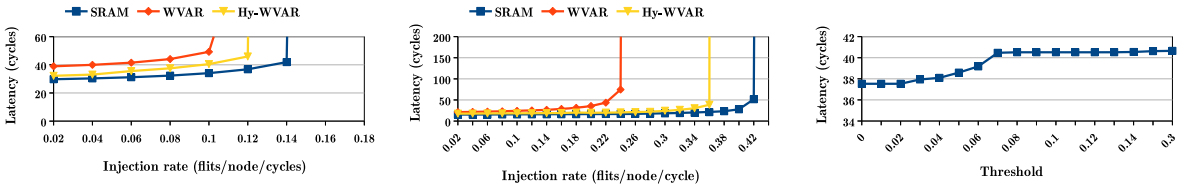


Fig. 8: Normalized EDP gains of proposed policies over SRAM baseline in 8×8 mesh network.



(a) Uniform random (b) Neighbor (c) Threshold analysis

Fig. 9: Performance Comparison with Synthetic Workloads in 8×8 mesh network.

packets the impact on lifetime improvement is maximum in the data network.

Total energy: The figure 6 shows the total energy for SRAM baseline and proposed policies. On account of a very less static energy of STT-RAM, we get a significant reduction in total energy in both the proposals. In particular, the total energy is reduced by 90% and 86% respectively in WVAR and Hy-WVAR. The writes in STT-RAM require more energy, which increases the dynamic energy, however, lesser static energy gives us overall energy savings.

Average Packet Latency: The write operation of STT-RAM takes 2 cycles, whereas for SRAM takes only 1 cycle [2]. Note that according to [20], we have used lower retention time in order to reduce write cycles. This impacts packet network latency and hence average packet latency. The figure 7 shows normalized latency in proposed policies with respect to SRAM. WVAR increases packet latency by 30% over SRAM. However, Hy-WVAR attempts to reduce this and achieves network latency of 12% over SRAM which is 14% improvement over WVAR.

EDP: Slight increase in network latency affects performance. However, the savings in static energy are significant and thus leads to an improved EDP. On average we get 93% and 90% gain in EDP over baseline SRAM [Figure 8].

Latency analysis: To study the impact of proposed policies on the packet latency, we performed simulations on an 8x8 network using different traffic patterns and varying the injection rate. Figure 9a shows performance comparison of

proposed policies with the baseline under $uniform-random$ synthetic workload. On account of the slow write speed of STT-RAM, the packets take slightly more time to reach the destinations. As expected, the latency of Hy-WVAR policy lies between the latency values for SRAM and WVAR and both the proposed policies reach saturation earlier than the baseline SRAM. WVAR increases latency by 28% on average, whereas Hy-WVAR shows latency improvement by 14% w.r t. WVAR.

In $neighbor$ synthetic workload, figure 9b, WVAR performance degrades and latency grows very high, which is 61% over SRAM. The hybrid policy, Hy-WVAR, shows 16% more latency with respect to SRAM which is 39% improvement over WVAR.

Effect of threshold value: The usage of SRAM based virtual channel in Hy-WVAR policy depends on the threshold used. Lower threshold value ensures more frequent writes in SRAM based VC. Figure 9c shows the effect on network latency using different threshold values under uniform-random traffic pattern. Higher threshold reduces number of writes in SRAM based virtual channel which results in more average packet latency. Thus, there is a trade-off between packet latency and power. If threshold is less, latency is less but it incurs more leakage power due to use of SRAM. On the other hand, a higher threshold saves leakage but incurs more packet delays.

V. Overhead Analysis

The counter for each virtual channel can be reset/truncated when any one of the counters in a virtual network reaches to its maximum value. This multiple times counter-reset allows us to use small counter such as $2B$. Therefore storage overhead for each virtual channel will be $2B$, which is total of $24B$ for given system parameters in table I. The number of buffers at any input port is 24, hence total size will be $384B$. The counter storage overhead is 6.25%. The area occupied by STT-RAM is one-fourth of the area consumed by SRAM memory cells. Hence, overall the buffer area, including counters, is reduced by 73.43% per port when we replace SRAM with WVAR based designs. In Hy-WVAR design, counter overhead is same as WVAR design. Here total area is reduced by 64.58% with respect to SRAM baseline design. The energy overhead of counter for 8×8 network is 22.7%. We have taken SRAM based counters and the counter energy overhead is mainly due to leakage energy. We have taken this overhead into account in our experiments. On the whole, as seen from figure 6, the overall energy is significantly lesser than SRAM baseline policy.

VI. Conclusion

NoC is the backbone of communication for today's chip multiprocessors and also a significant contributor to the power budget. Specifically, the buffers at the ports consume more power. This paper proposed to use NVM based STT-RAM buffers as they promise high density and low leakage. The major drawbacks of STT-RAM is their low write endurance which can affect the lifetime of the buffers. Towards this the paper proposed two policies to reduce write variation across buffers. The first policy WVAR uses pure STT-RAM buffers and maintains equal number of writes across all the VC within a VNet. This improves the lifetime by 3.4 times and reduces write variation to almost 0%. STT-RAM buffers incur more write latency and hence this affects the overall packet latency. We get around 30% increase in latency over baseline SRAM.

On account of using SRAM buffers in case of heavy traffic, the hybrid VC policy Hy-WVAR achieves 99.99% reduction in write variation and 24 times improvement in lifetime. This helps in improving the latency by 14% over WVAR policy. The reduced static energy consumption of NVM technology gives considerable reduction in total power consumption, leading to EDP gains of 90-93% in the proposed policies. Thus careful management of writes in NVM buffers can make them a viable replacement for power hungry SRAM buffers in on-chip interconnects.

References

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

[2] J. Zhan et al., "Hybrid drowsy SRAM and STT-RAM buffer designs for dark-silicon-aware noc," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 10, pp. 3041–3054, 2016.

[3] H. Jang et al., "A hybrid buffer design with STT-MRAM for on-chip interconnects," in *NoCS*. IEEE, 2012, pp. 193–200.

[4] Y. Huai, "Spin-Transfer Torque MRAM (STT-MRAM): Challenges and Prospects," *AAPPS bulletin*, vol. 18, no. 6, pp. 33–40, 2008.

[5] Y. B. Kim et al., "Bi-layered RRAM with unlimited endurance and extremely uniform switching," in *VLSIT*. IEEE, 2011, pp. 52–53.

[6] M. K. Qureshi et al., "Phase change memory: From devices to systems," *Synthesis Lectures on Computer Architecture*, vol. 6, no. 4, pp. 1–134, 2011.

[7] N. Binkert et al., "The Gem5 simulator," *SIGARCH Comput. Archit. News*, vol. 39, no. 2, pp. 1–7, Aug. 2011.

[8] N. Agarwal et al., "GARNET: A detailed on-chip network model inside a full-system simulator," in *ISPASS*. IEEE, 2009, pp. 33–42.

[9] D. Apalkov et al., "Spin-Transfer Torque Magnetic Random Access Memory (STT-MRAM)," *J. Emerg. Technol. Comput. Syst.*, vol. 9, no. 2, pp. 13:1–13:35, May 2013.

[10] P. Kundu, "On-die interconnects for next generation CMPs," in *Workshop on On-and Off-Chip Interconnection Networks for Multicore Systems (OCIN)*, 2006.

[11] T. Moscibroda and O. Mutlu, "A case for bufferless routing in on-chip networks," in *ACM SIGARCH Computer Architecture News*, vol. 37, no. 3. ACM, 2009, pp. 196–207.

[12] G. Michelogiannakis and W. J. Dally, "Elastic buffer flow control for on-chip networks," *IEEE Transactions on Computers*, vol. 62, no. 2, pp. 295–309, 2013.

[13] A. K. Kodi et al., "Adaptive inter-router links for low-power, area-efficient and reliable Network-on-Chip (NoC) architectures," in *ASP-DAC 2009*. IEEE, 2009, pp. 1–6.

[14] C. Li and P. Ampadu, "A compact low-power eDRAM-based NoC buffer," in *ISLPED*. IEEE, 2015, pp. 116–121.

[15] S. Agarwal and H. K. Kapoor, "Targeting Inter Set Write Variation to improve the lifetime of Non-volatile cache using fellow sets," in *VLSI-SoC*, Oct 2017, pp. 1–6.

[16] ——, "Reuse-Distance-Aware Write-Intensity Prediction of Dataless Entries for Energy-Efficient Hybrid Caches," *IEEE TVLSI*, vol. 26, no. 10, pp. 1881–1894, Oct 2018.

[17] X. Wu et al., "Hybrid Cache Architecture with Disparate Memory Technologies," in *Proceedings of the 36th Annual International Symposium on Computer Architecture (ISCA)*. ACM, 2009, pp. 34–45.

[18] J. Wang et al., "i2WAP: Improving non-volatile cache lifetime by reducing inter-and intra-set write variations," in *HPCA*. IEEE, 2013, pp. 234–245.

[19] 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. 423–428.

[20] A. Jog et al., "Cache Revive: architecting volatile STT-RAM caches for enhanced performance in CMPs," in *DAC*. ACM, 2012, pp. 243–252.

[21] C. Bienia and K. Li, "PARSEC 2.0: A new benchmark suite for Chip-Multiprocessors," in *Proceedings of the 5th Annual Workshop on Modeling, Benchmarking and Simulation*, vol. 2011, 2009.