A PURE STT-MRAM BUFFER DESIGN FOR HIGH-BANDWIDTH LOW-POWER ON-CHIP INTERCONNECTS

A Thesis

by

ROHAN KANSAL

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Chair of Committee,	Jiang Hu
Co-Chair of Committee,	Eun Jung Kim
Committee Member,	Gwan S. Choi
Head of Department,	Chanan Singh

August 2013

Major Subject: Computer Engineering

ABSTRACT

Network-on-Chip (NoC) is a de facto inter-core communication infrastructure for future Chip Multiprocessors (CMPs). NoC should be designed to provide both low latency and high bandwidth considering limited on-chip power and area budgets. The use of a high density and low leakage memory, Spin-Torque Transfer Magnetic RAM (STT-MRAM), in NoC routers has been proposed as it increases network throughput by providing more buffer capacities with the same die footprint. However, the inevitable use of SRAM to hide the long write latencies of STT-MRAM sacrifices buffer area and also wastes significant leakage and dynamic power in migrating flits between the disparate memories. In this thesis, the first NoC router designs that use only STT-MRAM is proposed. This allows for a much larger buffer space with the least power consumptions. To overcome the multi-cycle writes, a multi-banked STT-MRAM buffer is employed, which is a logically divided virtual channel where every incoming flit is seamlessly pipelined to each bank alternately every clock cycle simple latches inside the router links. Our STT-MRAM has aggressively reduced retention time, resulting in a significant reduction in latency and power overheads of write operations. We observe flit losses in our STT-MRAM buffer, and propose cost-efficient dynamic buffer refresh schemes to minimize unnecessary refreshes with minimum hardware overheads. Simulation results show that our STT-MRAM NoC router enhances the throughput by 21.6% and achieves 61% savings in dynamic power and 18% savings in total router power, respectively compared to a conventional SRAM based NoC router of same area.

To my family and friends

ACKNOWLEDGEMENTS

I would like to express my deep gratitude to my advisor, Dr. E. J. Kim for giving me an opportunity to work in the High Performance Computing (HPC) Lab under her. Her constant guidance and support always helped me move in the right direction in my research. I am really grateful to her for making my research experience a truly memorable one.

I would also like to thank Dr. Jiang Hu and Dr. Gwan Choi for agreeing to be on my thesis committee and providing me valuable feedback.

A special thanks to the HPC members who have worked with me on this, Dr. K.H. Yum, Hyunjun Jang and Rahul Boyapati. This work wouldn't have been possible without their valuable input and support.

My time at Texas A&M University was made enjoyable in large part due to many friends that became a part of my life. I am grateful for the time spent with roommates and friends.

I would also like to thank my family for their love, support and encouragement throughout my life, and for inspiring me to always reach for higher and better education.

TABLE OF CONTENTS

Page

LIST OF FIGURES

Page

LIST OF TABLES

Page

1. INTRODUCTION

The advance of CMOS technology has allowed us to achieve a high degree of parallelism through Chip Multiprocessors (CMPs) by fabricating multiple cores and cache banks in a single chip [25]. A switch-based Network-on-Chip (NoC) is a promising architecture orchestrating chip-wide communication towards future CMPs. NoC should be carefully designed due to the inherent constraints of the restricted power and area budgets in a chip. NoC consumes up to 28% of the chip power [14], and of the different components comprising on-chip interconnects, buffers are the largest leakage power consumers in NoC routers, consuming about 64% of the total router leakage power [4]. Buffers also consume significant dynamic power [34] [37], and this consumption increases rapidly as data flow rates increase [37]. Consequently, designing an innovative buffer structure plays a crucial role in architecting high performance and low power on-chip interconnects.

Spin-Torque Transfer Magnetic RAM (STT-MRAM) [23, 30, 32, 38] is considered as a promising next generation memory technology that can replace conventional RAMs due to its near-zero leakage power and high density. Utilizing STT-MRAM in NoC has significant merits since an on-chip router is able to provide larger input buffers under the same area budget compared to conventional SRAM. Communication in throughput oriented applications favor deeper buffers for higher bandwidth, resulting in a better system level performance with lesser power consumption. STT-MRAM also provides higher endurance compared to other memory technologies such as Phase Change

Memory (PCM) [17], Flash, and Memristor [13] which makes STT-MRAM a more viable solution as an on-chip memory that should tolerate highly frequent write accesses. An embedded DRAM (eDRAM) [3] has similar density, but higher leakage and refresh energy than STT-MRAM, and, moreover has scalability issues in sub-45nm process technology [18]. STT-MRAM is able to scale beyond 10nm technology [9]. However, the weaknesses of STT-MRAM, the long latency and high power consumption in write operations, should be properly addressed due to pre-requisites of fast accesses for on-chip memories.

Several researches have been performed to address these write speed and energy limitations of STT-MRAM in designing caches and NoC routers. A SRAM/MRAM hybrid cache with 3D stacking structure was proposed in [31]. A region based hybrid cache [35] with a small fast SRAM and a large slow MRAM mitigates the performance degradation and energy overheads. A SRAM/STT-MRAM hybrid buffer design [16] shows substantial throughput improvements across various workloads. In common, however, the inevitable use of SRAM to hide the multi-cycles writes of STT-MRAM sacrifices area, and also wastes significant leakage and dynamic power in migrating data between the disparate memories.

1.1 Thesis statement

In this thesis, we propose the first NoC router designs that use only STT-MRAM in buffers. By eliminating SRAM, it allows for a much larger buffer space with the least leakage and dynamic power consumptions. To hide the multi-cycle write latencies of STT-MRAM, we propose a novel pipelined input buffer design, a multi-banked STT-MRAM buffer, which is a logically divided virtual channel where every incoming flit is delivered to each bank alternately every clock cycle via a simple latch inside a router. Through this, we can avoid the degradation of performance while consuming the least area and power.

Our STT-MRAM has aggressively reduced retention time since relaxing the non-volatility of STT-MRAM leads to a lower write current and faster write operation [18] [32]. We note that the retention time needed in input buffers of NoC routers can be significantly reduced considering the short intra-router latency of a flit, defined by the time difference between arrival at the buffer and departure via a crossbar. Thus, we utilize the write latency reducing technique [32] to bring the write latency down to 2cycles by reducing the data retention time of an Magnetic Tunnel Junction (MTJ), a core unit of STT-MRAM, to 100ns which corresponds to 200cycles in 2GHz clock frequency. The data retention time indicates how long data can be retained safely in a non-volatile memory cell after being written. In most cases, the reduced retention time is enough to hold stored flits safely, however, we observe such cases that flits stay in STT-MRAM buffers longer than a given retention time, which lead the flits to become corrupted especially in a highly congested network. These lost flits may incur highly

noticeable performance losses or affect other network applications negatively especially in case the flit is control packet carrying critical cache coherence information. On average 78.7% of traffics are such single-flit control packets in PARSEC benchmarks [22]. This observation motivates us to devise cost-efficient dynamic buffer refresh schemes, the process by which cell's values are kept valid by triggering refreshes in a timely manner with minimum hardware overheads.

1.2 Thesis contribution

The main contributions of this thesis are as follows:

- We present a detailed qualitative analysis on design tradeoffs of an MTJ especially in terms of the MTJ's write performance, write power, and retention time, which are appropriate for implementing the optimal NoCs.
- We propose a novel pipelined input buffer design, a multi-banked STT-MRAM buffer, implemented entirely with STT-MRAM that delivers the optimal power saving and performance improvement.
- We propose cost-efficient dynamic buffer refresh schemes: a simple refresh scheme, and a global counter refresh scheme, which selectively triggers the refresh of a buffer in a power efficient manner to maintain the validity of flits.
- Detailed simulation results show that our proposed STTMRAM NoC router improves the overall network throughput by 21.6%and achieves 61%, and 18%power savings in the dynamic and total router power, respectively compared to the conventional SRAM-based NoC router.

1.3 Thesis organization

The rest of this thesis is organized as follows. Related previous work in this domain is discussed in Section 2, followed by the design characterization and tuning of STT-MRAM parameters and its data retention time model in Section 3. In Section 4, we explain the proposed STT-MRAM router architecture in detail. Section 5 presents experimental results and performance analysis, and finally Section 6 summarizes our work and makes conclusions.

2. RELATED WORK

2.1. STT-MRAM

STT-MRAM is a promising next generation technology in memory that exploits magneto-resistance for storing data. In STT-MRAM, each data bit is stored in a Magnetic Tunnel Junction (MTJ), the fundamental building block. An MTJ consists of three layers: two ferromagnetic layers and an Mg-O tunnel barrier layer in the middle as shown in Figure 1. Among them, the magnetization direction of the bottom layer is fixed. The spin of the electrons in the free layer relative to the fixed layer decides the content of the cell. The fixed layer polarizes the passing current and influences the direction of spin of the top layer. When the spin polarity of the top layer is same as that of the bottom layer, the cell acts as a low resistance causing the state of the cell to store a one. On the other hand, an anti-parallel state of the MTJ creates a high resistance path and thus stores a zero. Figure 1 shows the parallel or high state in (a) and the anti-parallel or low state in (b). A STT-MRAM cell is made of 1T-1MTJ module, which corresponds to a 1 NMOS transistor coupled with a MTJ cell as shown in Figure 1.

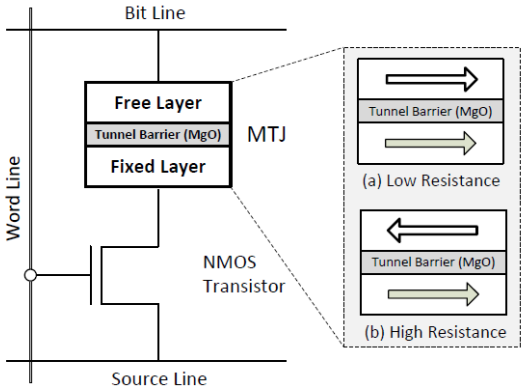

Figure 1: STT-MRAM cell structure

2.2. STT-MRAM in processors and memories

Use of STT-MRAM has been explored in the past, but mostly with off-chip memories or on-chip caches. Mishra et al [22] mitigated the long write latency for STT-MRAM based caches in a 3D CMP by postponing cache accesses to banks that were busy. Jog [17] and Smullen et al [29] both proposed reducing the retention time for on-chip caches to lower the write latency and energy consumption for L2 caches. While Jog et al [17] adjusted the various MTJ parameters; Smullen [29] tailored them for on—chip cache usage. Before that, Sun et al [30] proposed stacking MRAM for L2 caches on top of CMPs and reduce excess overheads via read pre-emptive write buffers and hybrid caches using both STT-MRAM and SRAM. In [10], Guo et al address issues concerning STT-MRAM design and their use in power efficient CMP systems. Most recently, Sun et al. [31] deployed STT-MRAM in L1 caches as well as L2 caches in CMP systems. They improved the write performance of STTMRAM in L1 caches by adjusting the retention time of STT-MRAM in a ultra low level, while providing a power-efficient refresh scheme to prevent data corruption in L1 caches.

2.3. Buffer design in NOC routers

Designing efficient input buffers in NoC is critical in improving both network performance and power efficiency. The dynamic virtual channel regulator (ViChaR) [23] allows for dynamic management of buffer resources and additional dynamic allocation of VCs per output port, which helps improve overall network throughput. A most recent work [15] that adopts STT-MRAM in designing NoC input buffers suggests a new hybrid design combining both SRAMand STT-MRAM to effectively hide the long write

latency of STT-MRAM. While it shows substantial throughput improvements, it fails tomake the best use of the low power features of STT-MRAM due to the high leakage power of SRAM and the frequent migration operations from SRAM to STT-MRAM, thus wasting a significant amount of extra dynamic power. There also have been attempts to provide alternative buffer storages to minimize the power and area overheads. Kodi et al. [20] explored the use of repeaters existing along the inter-router links as potential buffer storages by triggering a control signal at high network loads when there are no more buffers in the router. The inter-router dual function links lead to a reduction in power consumption and area overhead without significant loss in performance.

3. STT-MRAM BUFFER DESIGN CONSIDERATION

In this section, we describe important design tradeoffs of an MTJ cell in terms of switching time(i.e. write latency), switching current(i.e. write power), and data retention time, and details the flit dropping issues caused by the insufficient retention time of STT-MRAM, and propose cost-efficient dynamic refresh schemes.

3.1. Tradeoffs in MTJ designs

a) **Retention time**: The non-volatility of an MTJ is quantitatively measured by the data retention time, which is the maximum time duration for which data can be stored in the MTJ. It can also be measured as the MTJ switching time when the switching current is zero. The data retention time, T_{ret} , of an MTJ cell is defined as follows [27]

$$T_{ret} = 1ns \cdot e^{\Delta}$$

Δ is the thermal factor that estimates the thermal stability of an MTJ module. The thermal factor depends on the saturation magnetization (M_s), the in-plane anisotropy field (H_k), volume of an MTJ cell (V), and the working temperature (T) as follows [8].

$$\Delta \propto \frac{M_s H_k V}{T}$$

We decrease the thermal factor by reducing M_s and H_k, resulting in the reduced retention time [29].

b) **Switching current**: In a precessional switching mode [26] where an MTJ switching time (τ) is short (<3ns), the required current density, $J_c(\tau)$, is determined as follows.

$$J_{c,PS}(\tau) = J_{c0} + \frac{C}{\tau^{\gamma}}$$

J_{c0} is a switching threshold current density that also depends on M_s and H_k [37] like the thermal factor above. C is a constant affected by the initial angle between the magnetization vector of the free layer and the easy axis. Reducing the retention time causes the thermal factor to decrease, which reduces M_s and H_k, and eventually decreases J_{c0}. Therefore, with smaller J_{c0}, we can achieve shorter switching time with the reduced current density. In our case, specifically, reducing the retention time up to 100ns allows for smaller J_{c0}, and through this, 2 cycles of write latency, corresponding to 1ns in 2GHz clock frequency, is achieved with 71.35mA of switching current as shown in Figure 2, which can be provided by $31.3F^2$ of STT-MRAM cell size. These results are based on the simulation with the PTM model [2] under 32nm technology. We also obtained raw experimental data and did curve fitting using the MTJ device equations in [26] for different retention times to get Figure 2.

By aggressively relaxing the retention time to only 100 ns, we get a safe operating with 1ns write latency with the switching current of 71.35mA. Similarly, Sun et al. [31] model MTJ cell with switching time reduced to 1ns where the write energy is further reduced down from 1.86pJ to 0.32pJ by decreasing the MTJ's data retention time from 6.7x106 years to 4.27 years.

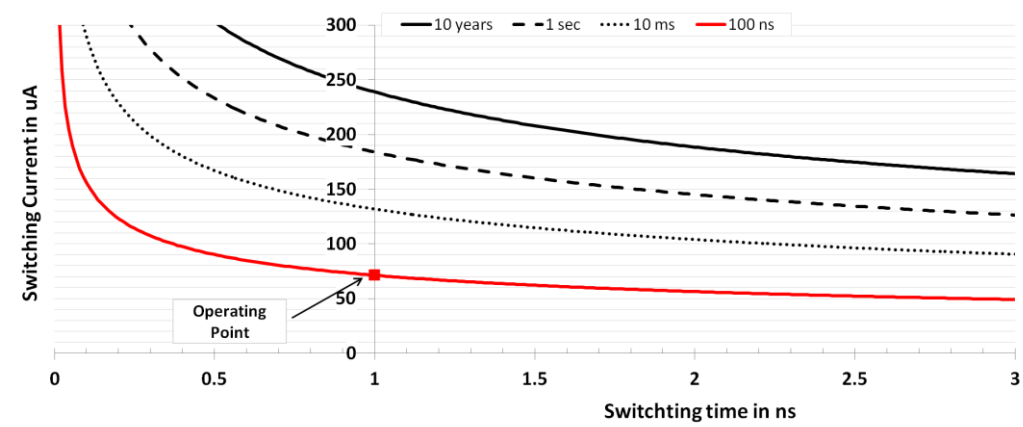

Figure 2: Switching current vs switching time for different Retention times

It is noted that to achieve the same write performance as SRAM, writing to STT-MRAM must be done in a single cycle that corresponds to 0.5ns in 2GHz clock frequency. Such sub nanosecond switching times are possible [7], but require rather strong currents and are far from the optimal efficiency [32]. It is also impractical to reduce the write latency of STT-MRAM as fast as SRAM because it may result in a significant amount of flit dropping due to the huge sacrifice of the retention time.

c) Area: We refer to ITRS 2009 projections [14] and the model used by Guo and Ipek [10] to model the STT-MRAM area in our simulations. The size of 1T-1MTJ according to this model is $30F^2$, where F is the process technology. For 32nm technology, the size of an SRAM cell is approximately $146F^2$, and thus is 4.8 times the area of a STT-MRAM cell. It is because of this difference in density that 4 cells of STT-MRAM could be fit in 1 SRAM cell area budget. An area of the STT-MRAM is primarily determined by the current that is required to switch the states of the free layer (parallel → anti-parallel and vice versa). The maximum write current decides the transistor size [10]

which dominates the STT-MRAM cell size, since the MTJ cell is much smaller in comparison. STT-MRAM cell area is able to be derived as follows [17].

$$Area_{cell} = 3(\frac{W}{L} + 1)(F^2)$$

where W and L represent the channel width and length of the NMOS transistor, respectively.

d) Energy: Dynamic write energy consumption of STTMRAM cell is calculated by the following equation:

$$E = \frac{V^2 T_s}{R_{MTJ}}$$

where V is power supply voltage and R_{MTJ} is the resistance of an MTJ cell [38]. Since the write energy is proportional to the switching time, T_s, the energy consumption reduces as the write latency decreases. In our case, a switching current of 71.35mA and R_{MTJ} of 21.02kW leads to a write energy of 13.7pJ/flit for 1 ns switching time. For reads, our operating point is at 28.12mA leading to a 2.7pJ/flit read energy. These read/write metrics are used in Table 2 in Section 5.

3.2. The non-volatility relaxed STT-MRAM buffer

As the data retention time of STT-MRAM gets decreased, maintaining the validity of flits in STT-MRAM buffer gets harder as network becomes congested. In this subsection, we quantitatively examine flits which are dropped by staying in a buffer

longer than the given retention time, identify various factors affecting the flit dropping, and propose cost-efficient dynamic buffer refresh schemes preventing such flit losses.

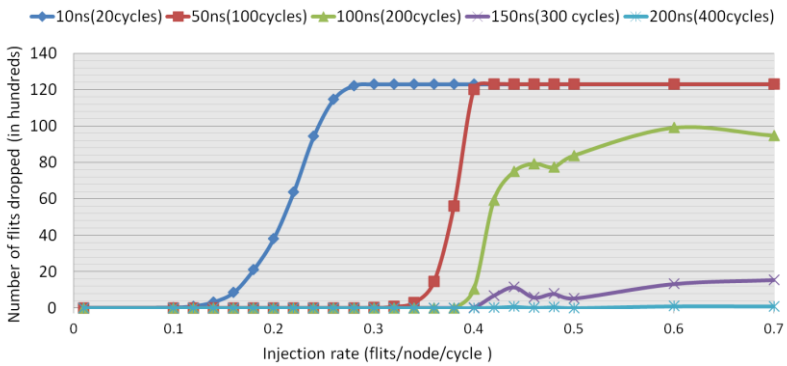

Figure 3: Flits dropped for different retention times

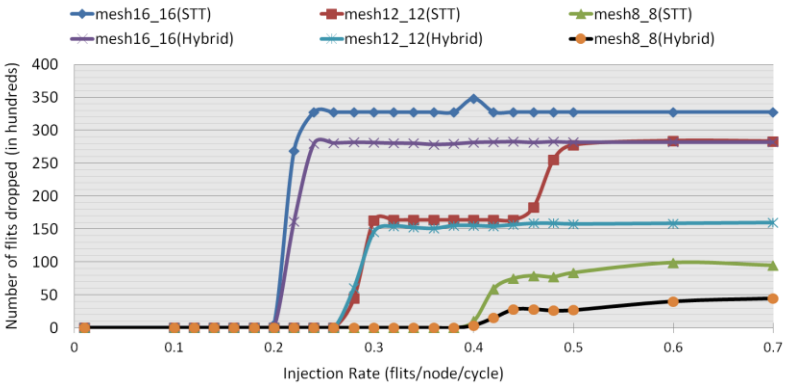

Figure 4: Flits dropped for different mesh sizes

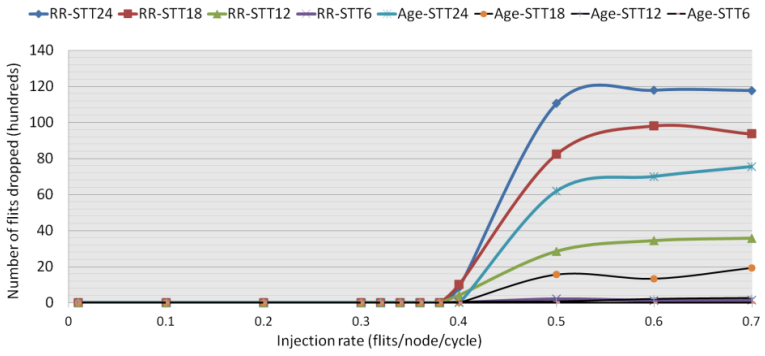

Figure 5: Flits dropped for different SA and buffer sizes

3.2.1. Various factors affecting flit dropping

a) Different buffer depths: To examine how the decreased retention time affects the validity of flits in STT-MRAM buffers, we conduct experiments with traces from PARSEC benchmark [1] as shown in Figure 6. All results are measured under the same area budget, 6 SRAM slots per VC, for input buffers. For comparison purposes, we use two different input buffer configurations with the same area budget: the Hybrid buffer [15] having 3 SRAM and 12 STT-MRAM per VC, and the STT-MRAM buffer having 18 STT-MRAM per VC. Figure 6 shows the maximum cycles of the oldest flit staying in buffers across 9 benchmark applications. It shows that the STT-MRAM buffer has an average of 46.5% higher intra-router latency than that of the Hybrid buffer mainly due to the deeper buffer size. Consequently, the STT-MRAM buffer causes 2.6 times more flits to be dropped on average than that of the Hybrid buffer.

b) Different retention times & c) Network scales: It is also noteworthy to consider the experimental results of Figure 3 showing that even under a certain level of sufficiently long retention time of STT-MRAM, as the packet injection rates increase, that is, on-chip network gets more congested by several bottleneck factors e.g. Head-of-Line (HoL) blocking or deadlock, flits get still dropped. Also, as the network scale increases, more flits tend to get dropped as shown in Figure 4.

d) Different switch arbitration schemes: Figure 5 shows the different number of flits dropped when applying two different switch arbitration schemes, RR and Age-based SA, under different buffer sizes per VC. Across all buffer sizes listed in Figure 5, the number

of dropped flits of Age- 4 based SA is lesser compared to RR by approximately 73.6% on average. This is due to the RR's inherent characteristics of choosing flits in a VC without considering their relative age, thus it is not effective in reducing the number of dropped flits. On the other hand, in Age-based SA, oldest flits are always chosen over others, thus significantly contributing to lower the possibility of flit dropping.

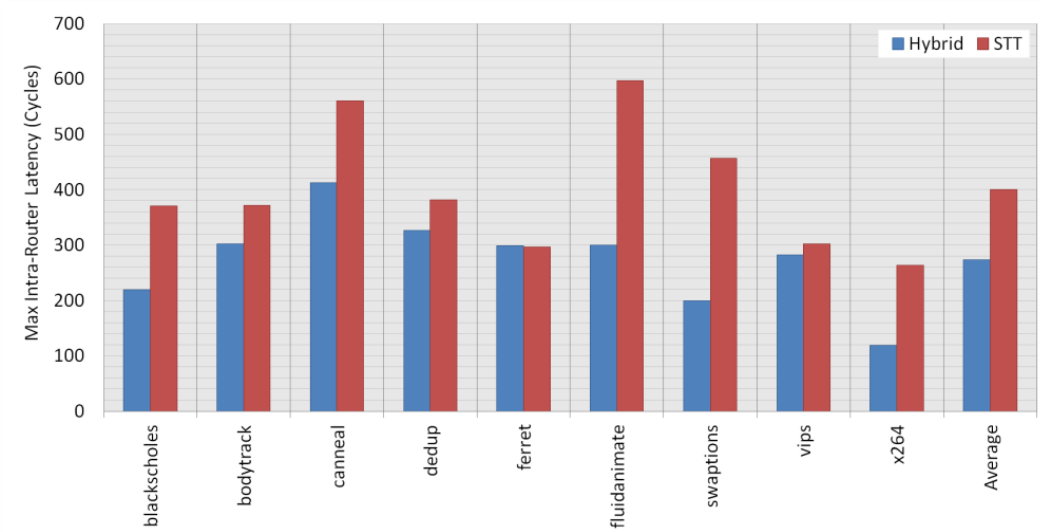

Figure 6: Maximum intra-router latency for PARSEC

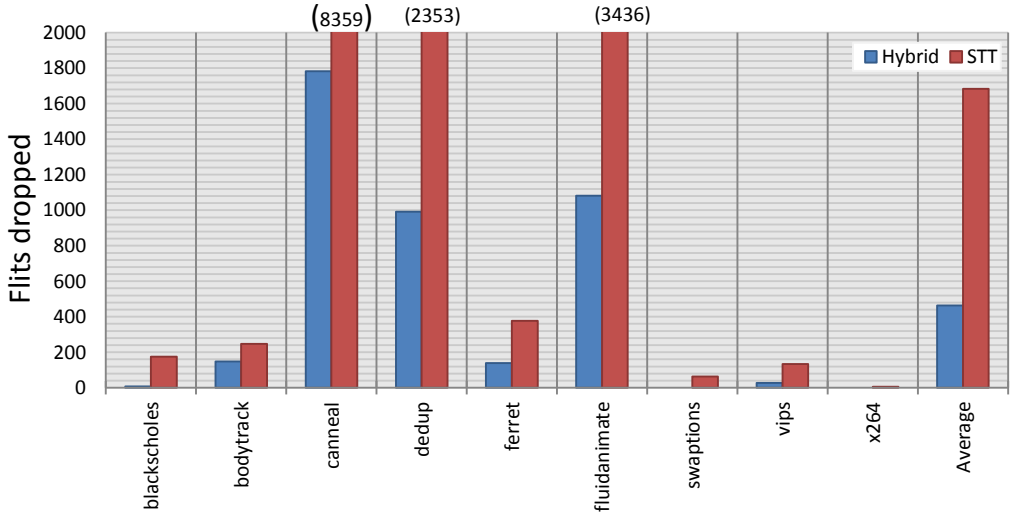

Figure 7: Flits dropped with STT retention time of 100ns

These kinds of packet/flit dropping issues have been widely studied in macro and Internet networks. In essence, a source router waits for an acknowledgement from a destination router, based on which the source router either continues the transmission or re-transmits the unsuccessful packet. However, for on-chip networks, this wastes a lot of valuable buffer spaces, and retransmission consumes additional network bandwidth. We thus propose cost-efficient buffer refresh schemes to eliminate flit dropping with the least number of refreshes.

3.2.2. Dynamic buffer refresh schemes for data retention

Due to the retention time constraints of STT-MRAM buffers, flits that exceed the volatility of the MRAM cells are at risk of being dropped or corrupted. This leads to a design decision to either 'refresh' the buffer slots leading to increased dynamic energies, or to drop the flit and resend it from the source router.

Packet dropping has been widely studied and researched in macro and internet networks. In essence, the source router waits for an acknowledgement from the destination router, based on which it either continues transmission or re-transmits the unsuccessful packet. However, for on-chip networks, this wastes a lot of valuable buffer space and retransmission consumes additional network bandwidth, rendering this approach inefficient for them. We thus propose mechanisms to intelligently refresh flits to eliminate packet dropping using the least number of refreshes.

Each of these mechanisms requires a 2 flit deep temporal storage buffer slot for each physical channel of the router. When a flit in any of the virtual channels in that physical

channel is marked for refresh, the temporal buffer is used as a small read buffer to compensate for the 2 cycle write latency for STT-RAM. The refreshes are pipelined to allow consecutive refreshes. We observe that any 2 flits in the same physical channel cannot arrive in the same cycle, and hence cannot also expire in the same cycle. This means, at a given time, all flits in all the virtual channels will have unique ages before refresh, or in other words, no two flits in a PC will have the same age. This facilitates us to keep only a single temporal buffer per physical channel to address the refreshing needs for all its virtual channels. Please note, this means we only refresh one flit per cycle per PC.

The simplest way would be keep track of the ages of all the flits currently residing in the buffers. For a 200 cycle retention time, 8 bit counters for each buffer slot that are updated every cycle would lead to prohibitive storage and energy overheads. It would however, results in least refreshes and thus can act as an important benchmark to which we compare our other schemes that are more cost –effective.

1) Simple refresh scheme is a straight-forward refresh scheduling algorithm, which simply forces a refresh operation to be performed as soon as a flit queued in the header of a VC reaches a certain refresh threshold, REFth. The REFth is predefined statically, expressed as clock cycles i.e. 100cycles in 2GHz clock frequency, and checked periodically by a refresh logic shown in Figure 10. Based on the FIFO properties of the buffers, the forefront flit is always the oldest one in a VC. Accordingly, this scheme guarantees that every subsequent flit stored in the same VC is also refreshed in a

proactive way within the retention time interval of STT-MRAM, thus no flits are dropped in any case. However, the simple refresh scheme unnecessarily wastes lots of dynamic power in a high network load where lots of flits circulate between routers every cycle .This is because it considers only the age of the forefront flit inside a VC, thus refreshing a group of the newly arrived flits too early. To minimize such a side effect of the simple refresh, we propose a more fine-grained refresh policy that triggers refresh operations per flit in a VC.

2) Global Counter (GC) refresh scheme: The motivation is to estimate the age of a flit residing in the buffer by using reduced resolution. It is also preferable to have minimum updates to this to reduce any read/write energies. An n-bit global counter (GC_n) acts as a wall-clock trigger for the router. This counter loops over after 200 cycles (Retention time) and increments after every $\frac{(\boldsymbol{Retention\ Time})}{\boldsymbol{2^n}}$ clock.

As shown in Figure 8, each buffer slot is associated with its own n bit Arrival Counter (AC_i). When a flit arrives in the buffer, the value of the GC_n is copied to its AC_i. At each update of GC_n, the AC values for all slots are compared. Flits whose AC_i value is one more than the current GC_n value are flagged for refresh. In case of multiple matches, the flit closer to head of the VC is given precedence in that cycle. The AC_i values are unchanged even after refreshes.

In order to ensure no flits are dropped, it is required that all flits inside a VC can be refreshed before the first one re-expires. This leads to the following condition on n:

$$\frac{(\boldsymbol{Retention\ Time})}{\boldsymbol{2^n}} \geq Buffer_{size} \quad \textbf{.... (1)}$$

Higher value of 'n' means tighter bounds for the first refresh and hence lesser refreshes in total.

For e.g. n = 2 means the global counter is updated every 200/2*2 = 50 clocks. At GC = 1, all flits with AC = 2 are refreshed. This is equivalent to refreshing flits with ages 100-150. However, all subsequent refreshes happen after exactly 200 cycles since the AC values are not changed.

With n = 3, the first refresh happens at age 150-175, and hence leads to even lesser refreshes. For a VC size of 18 (SRAM6 budget), n=3 is the maximum resolution, after which, condition (1) fails. Similarly, for a VC size of 12, n = 2, 3, 4 are possible. N = 4 in this case leads to first refresh at ages 174-187.

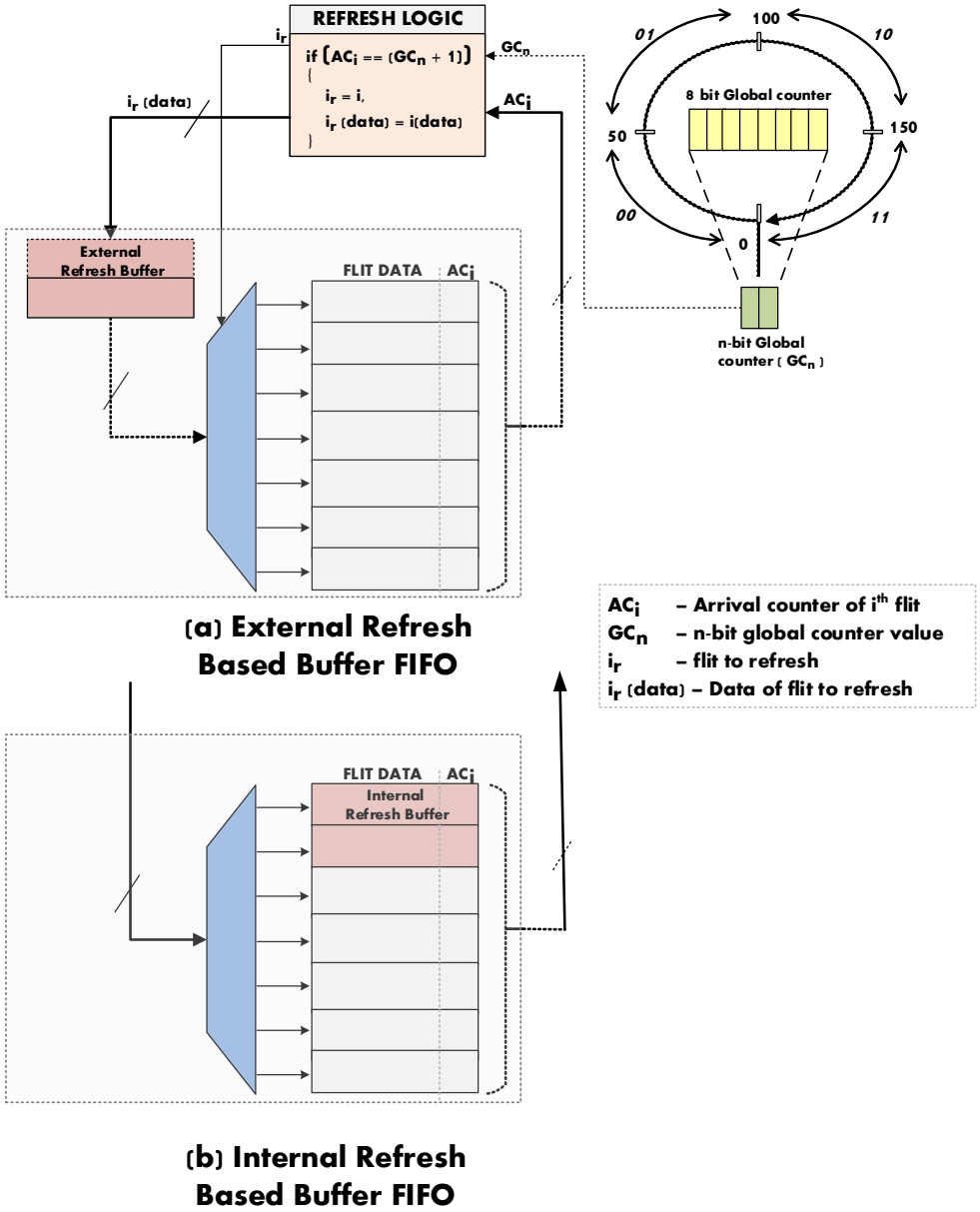

Figure 8: 2 bit global counter based refresh logic

(a) with external buffer and (b) with internal buffer

In this way, we can save significant power consumption associated with refresh operations compared to the aforementioned simple refresh or periodic DRAM-style refresh [29], where refresh operations are performed for all STT-MRAM cache blocks in

sequence regardless of its data contents, thus introducing many unnecessary refresh operations. The total refresh power consumed by the GC refresh scheme is reduced by up to 39.6% compared to the simple refresh scheme, which will be detailed in Section 5.3.

3.2.3. Internal refresh buffer (IRB)

Till now, we assume that the refresh operation is performed based on an External Refresh Buffer (ERB) as shown in Figure 8(a). One common drawback for these refresh mechanisms is that every refresh is basically requires two writes, one into the temporal buffer and one to the original buffer slot of the flit. The second mechanism we propose tries to reduce the number of writes required to refresh a flit, thereby saving write energy.

The intuition is that instead of writing into a temporal buffer (or the external refresh buffer) and writing back on each refresh, moving the flit into a new buffer slot would suffice. So instead of having a 2-flit deep storage that acts as a temporal buffer, we can have two internal refresh buffer (IRB) slots in each virtual channel as shown in figure 8(b). These slots are hidden from the upstream router to prevent the VA stage in the upstream router from allocating these slots to flits. When a flit is about to expire it can be refreshed by just ‘moving’ it to one of the empty slots. A single write would thus suffice as a refresh. Since the virtual channels are managed like circular buffers these empty slots keep progressing through the buffer as flits get refreshed. This is possible because the flits arrive into the buffer in order and hence will expire in order.

Since the IRB slots can reside between the head and tail pointer of the VC, a separate pointer is maintained to track them. The observation that the dummy slots generally stay together as flits are progressively refreshed, allow us to do this using only one pointer. The VC is read from head to tail, while 2 slots starting from the empty slot pointer are skipped.

However, there are a few instances when we observe that flits are scheduled for their second refresh before the first refresh of a new flit. This leads to an out-of order refreshing in a VC, resulting in in-correct ordering of pointers. Such cases are rare and are tackled by using these empty slots as temporal buffers. That is, the flits are written back to their original buffer slots, maintaining the in-order nature of the circular buffer.

4. STT-MRAM ROUTER ARCHITECTURE

Here we first describe the architecture of our generic baseline router and its input buffer architecture. We will then present our pure STT-MRAM based router in detail.. The key design goal of our proposed scheme is to enable flits to be written into buffers every cycle without incurring any additional timing delay.

4.1. Generic baseline router architecture

Figure 9 shows the generic baseline router architecture. This is based on L.Peh and Dally's [25] state of the art speculative architecture. This has a 2 stage pipeline, where the first cycle comprises of the Routing Computation (RC), VC allocation (VA) and Switch Arbitration (SA). The Switch Traversal (ST) takes the complete second cycle. The RC stage is removed from the critical path by adopting Galles' [9] look-ahead routing scheme. For an incoming flit, this generates the routing information of the downstream router prior to buffer write.

We use flit level wormhole switching [6] with credit based VC flow control as proposed by W.J. Dally in [5]. Each router consists of multiple VC's per input port. The number of input port is based on the topology used. We mostly use a 8x8 2D mesh but do a sensitivity analysis for other topologies as well. The credit based flow control [5] prevents loss of flits due to buffer overflows by providing an infrastructure to monitor back-pressure from downstream to upstream routers to control rate of flit transmissions.

Limited area and power constraints for on – chip buffers and ultra low latency requirements, NoC router buffers make use of very simple structures. Virtual channel based routers model buffers as parallel FIFOs where each virtual channel is essentially one FIFO. There are 'p' input ports, where each input port has 'v' FIFOs corresponding to 'v' virtual channels, and each virtual channel FIFO is 'k' flits deep. 'v' and 'k' for on-chip networks are well below macro networks to minimize area overheads. Due to these constraints and pre-requisites of low latencies, VC FIFOs are structured using a parallel FIFO design as depicted in Figure 10. The parallel structure has significant advantages over the serial implementation since flits do not have to traverse all buffer entries before leaving the VC [35]. Though this is faster, it does require a separate more complex control logic to manage the head and the tail pointers and maintain the FIFO properties. The head essentially corresponds to the read pointer and the tail to the write pointer. These are updated at every read/write inside the VC buffer.

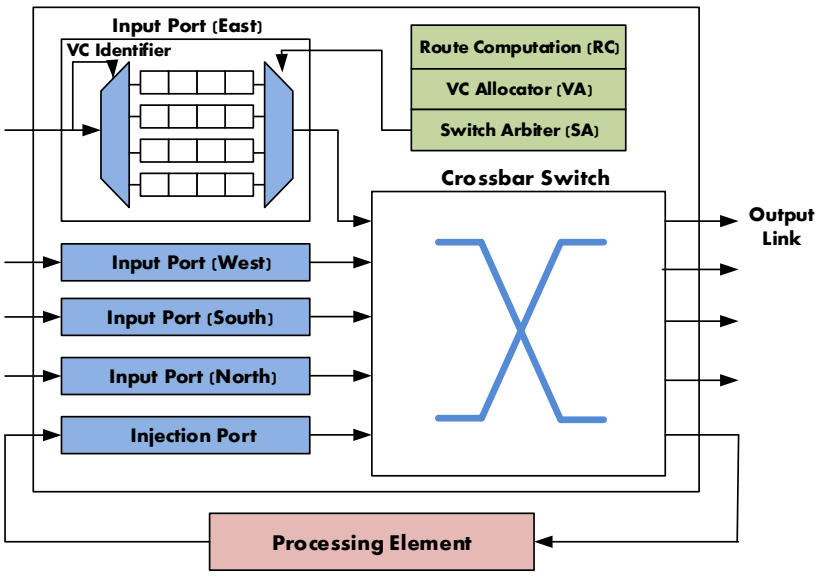

Figure 9: Generic baseline router architecture

The read and write pointers in the parallel FIFO registers regulate the operations of an input de-multiplexer and an output multiplexer, and the two pointers are controlled by a VC control logic, which generates proper read and write pointer values and status signals.

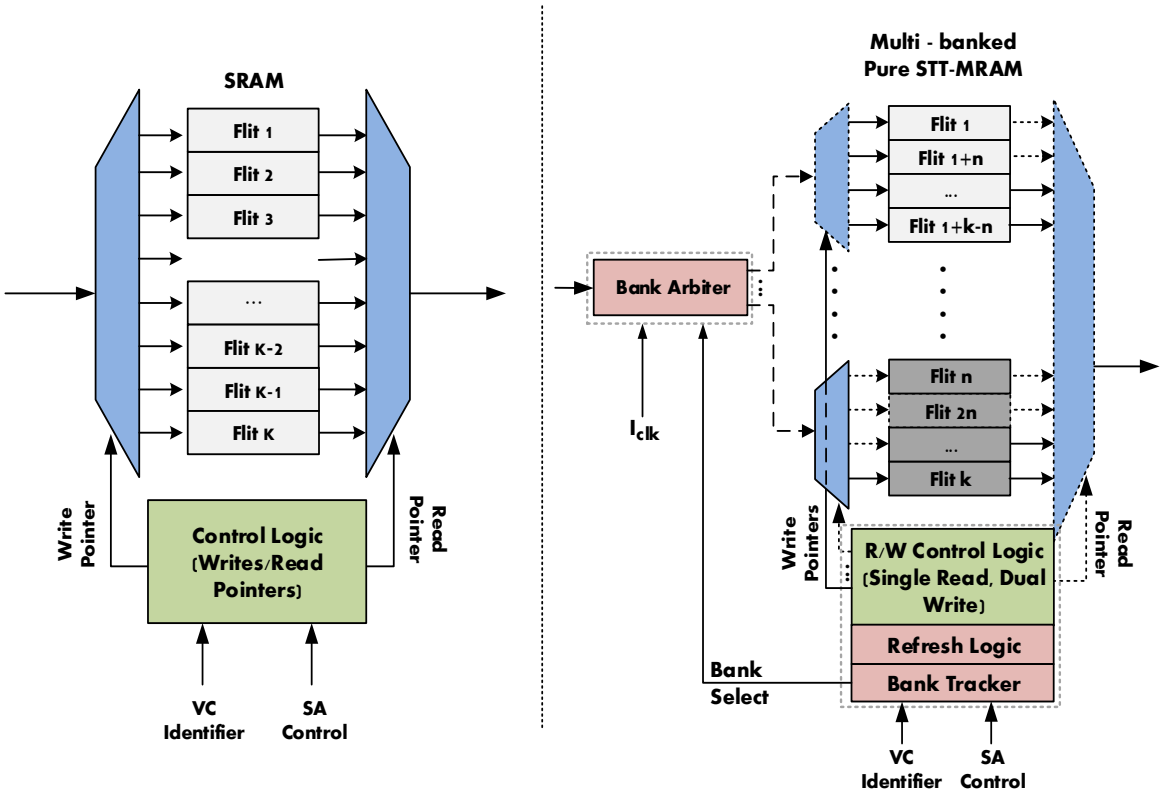

Figure 10: Generic SRAM input buffer and STT-MRAM banked input buffer

The read and the write pointers control the FIFO properties of the circular buffer. The read pointer is essentially the head of the buffer. It points to the flit that is to be read and is thus subsequently forwarded to the crossbar input port. The write pointer on the other hand is analogous to the tail pointer of the buffer. It points towards the slot inside the FIFO buffer where the new flits needs to be written to. After every write, this tail pointer

is incremented by 1. Similarly, the head pointer is also incremented after every read. At all times, all flits reside between the head and the tail pointer. The buffer is deemed empty if the read write pointers overlap after a read, and is full if they overlap after a write.

4.2. Design of STT-MRAM router

The generic baseline architecture of a router is modified to use STT-MRAM buffers instead of SRAM. The architectural changes for a single virtual channel are shown in Figure 10. To hide the multi-cycle write latency, each input buffer parallel FIFO is divided into a dual ported dual banked buffer. A bank arbiter splits the incoming flits to odd and even banks, while the control logic updates the relevant read / write pointers. We will discuss each of these elements in detail.

4.2.1. Dual banked STT-MRAM buffer

To hide the 2 cycles write delay of the STT-MRAM buffer, we logically divide each VC into two banks where every incoming flit is seamlessly pipelined to each bank alternately every clock cycle via a simple latch, which is deployed on a communication link inside a router. To clearly illustrate our logic, we to the two banks as an Odd and an even bank respectively, and every incoming flit from upstream routers as an Odd or an Even flit as shown in the left-hand side of the Figure 11(a). Every odd numbered flit is sent to the Odd bank of a downstream router, and likewise, even numbered flit to the even bank through a Dual Bank Arbiter (DBA) in Figure 11(a) which has one input port and two split output ports to each bank.

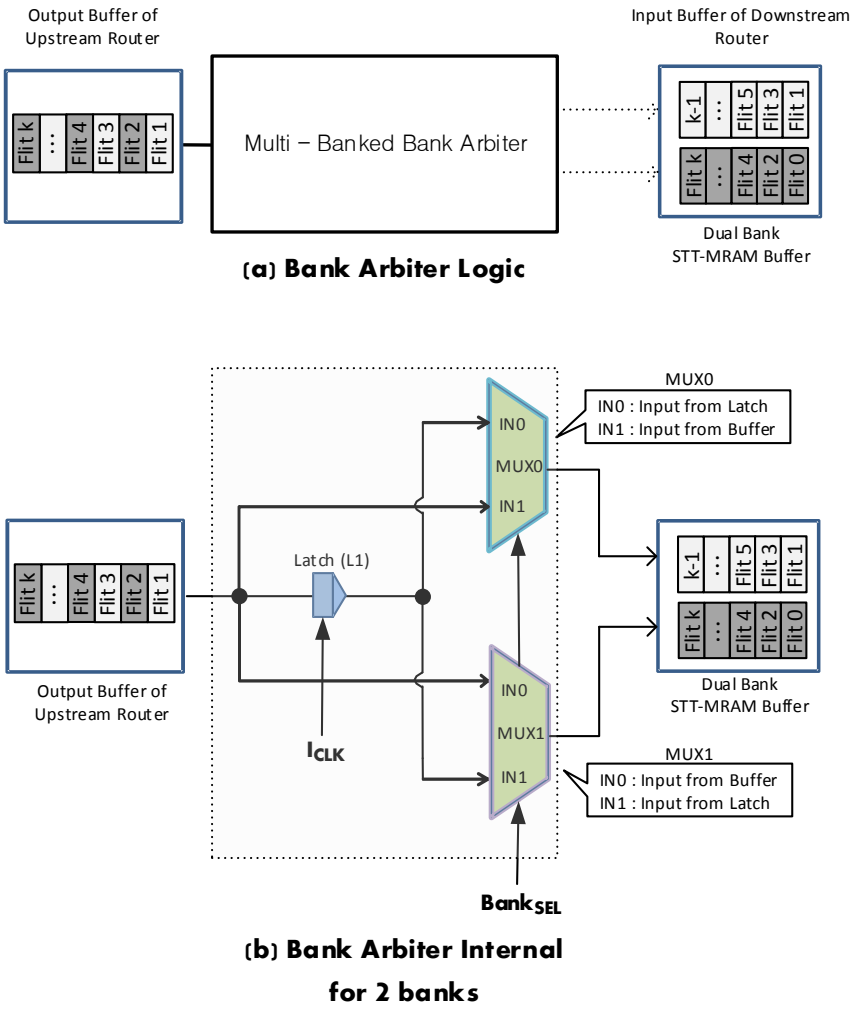

Figure 11: Bank arbiter logic and internal (2 banks)

4.2.2. Dual banked arbiter (b-ARB)

The DBA splits the incoming stream of flits to even and odd banks and pipelines their write cycles, as shown in Figure 11(a). This is done in the link traversal stage of the router pipeline and hence does not take an additional cycle.

Fig 11(b) shows the internal logic of the DBA. It makes use of a simple latch as a temporary link buffer. The input and the output of the latches are connected to MUX0

and MUX1 in an interleaved way to allow pipelining of the write cycles. The output of MUX0 drives the write port of the odd bank and MUX1 drives the write port of the even bank.

Dual banked STT-MRAM buffer working example

Figure 12-14 show an example of the data flow of three incoming flits at consecutive three clock cycles. Initially, the control of each MUX, denoted as MUX0 and MUX1, is assumed be set to 0, and all VCs are empty. It is also assumed that the interconnect clock period is long enough to satisfy the setup and hold constraints of a simple CMOS MUX.

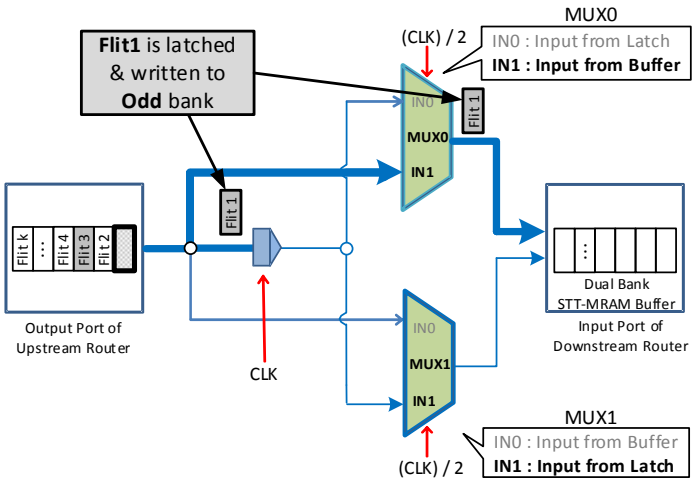

Figure 12: STT-MRAM buffer working example - cycle 0

(a) **Cycle 0**: In the first cycle, the input signal of both MUXes is set to 1 (IN1). This is the first write cycle for an incoming flit, Flit1. The Flit1 is sent to the Odd bank input buffer of the downstream router through IN1 ofMUX0, and at the same time, the Flit1 is latched at simple latch. Figure 12 shows this.

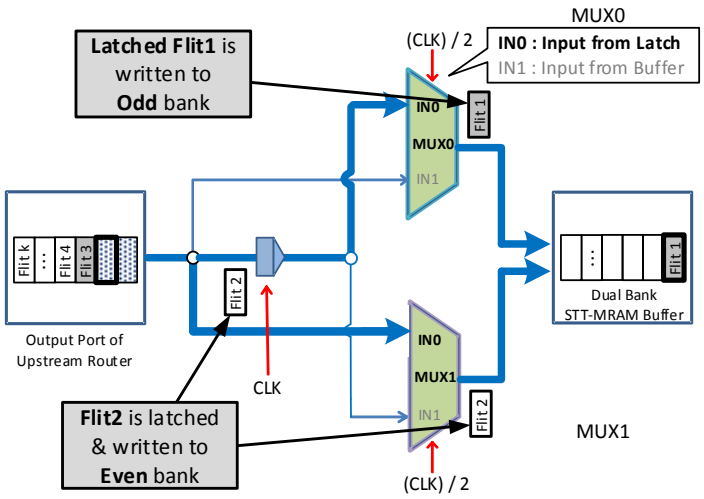

Figure 13: STT-MRAM buffer working example - cycle 1

(b) **Cycle 1**: In the second cycle, the input signal of both MUXes has changed to 0 (IN0), and this is the first write cycle for a subsequent incoming flit, Flit2, to the Even bank input buffer. While the Flit2 is transferred to the Even bank through IN0 of MUX1, it is simultaneously latched at the simple latch. Since the output ofMUX0 is still Flit1 which was previously latched, the Flit1 is dispatched from the latch to its previous target bank i.e. Odd bank of the input buffer, and thus completing its second write cycle. This is depicted on Figure 13.

(c) **Cycle 2**: In the third cycle, the input signal of both MUXes is switched back to 1 (IN1). Like the preceding logic, this is the second write cycle of the previous Flit2 from the latch to the Even bank, and the first write cycle for the subsequent incoming flit, Flit3. Figure 14 shows this.

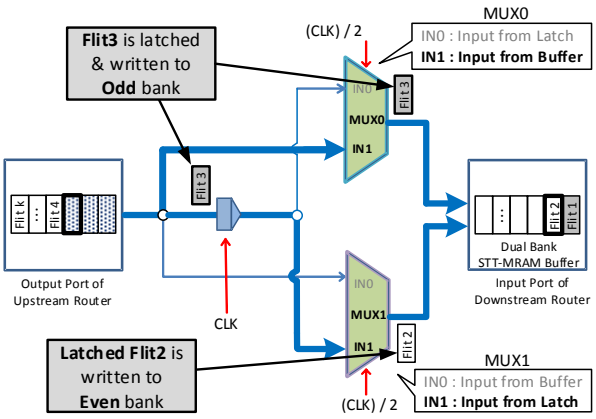

Figure 14: STT-MRAM buffer working example - cycle 2

It can be seen, that in 3 cycles, 2 flits have already been written on to the dual bank STT-MRAM Buffer. The throughput at steady state will thus be equal to 1, where the buffer is effectively updated every cycle. It is noted that throughout the whole processes, a bank availability tracker directs the incoming flit to either Odd or Even bank per each write to ensure that incoming flits are contiguously placed inside the designated VC without leaving unused buffer slots in a bank.

Timing diagram: Figure 15 shows a timing diagram corresponding to the dual bank STT-MRAM buffer working example shown in Figure 12-14. The I_{clk} indicates interconnect clock synchronized with the simple latch, and the M_{clk} indicates MUX clock synchronized with each MUX. The two MUXes are controlled by a common select signal which changes once every cycle, and has a clock of half the frequency of the I_{clk}. The input signal of MUX0 and MUX1 is alternated on a cycle-by-cycle basis as shown in cycle 0 through 5. During cycle 0, half of the Flit1 is first written to Odd bank, and

then at the very next cycle Flit1 completes its write process. The consecutive arriving flits go through the same pattern of write process to each different bank.

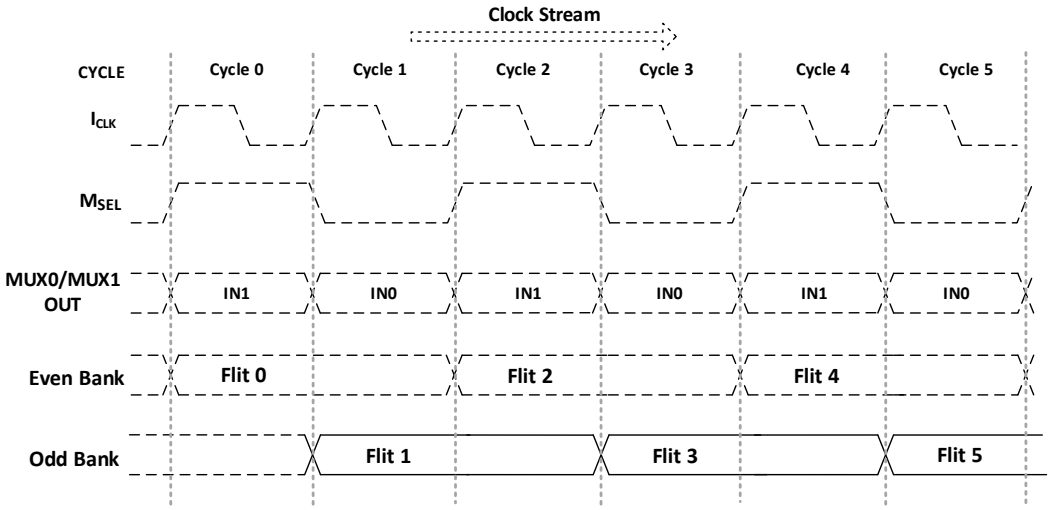

Figure 15: STT-MRAM buffer working example – timing

The control logic updates the relevant read and write pointers for each of the above cycles. A bank availability tracker keeps track of the available bank. It directs the incoming flits to either even or odd bank for each write to ensure flits are contiguously placed inside the designated VC without leaving unused buffer slots in a bank.

4.2.3. Control logic and Bank Availability Tracker (BAT)

The control logic monitors all the pointers for each VC in a router. Unlike a conventional SRAM buffer that has a single read and write pointer, the STT-MRAM buffer scheme requires 2 write pointers, 1 read and 1 refresh pointer. Fig 16 shows the parallel FIFO as a circular buffer with all the pointers and their updates.

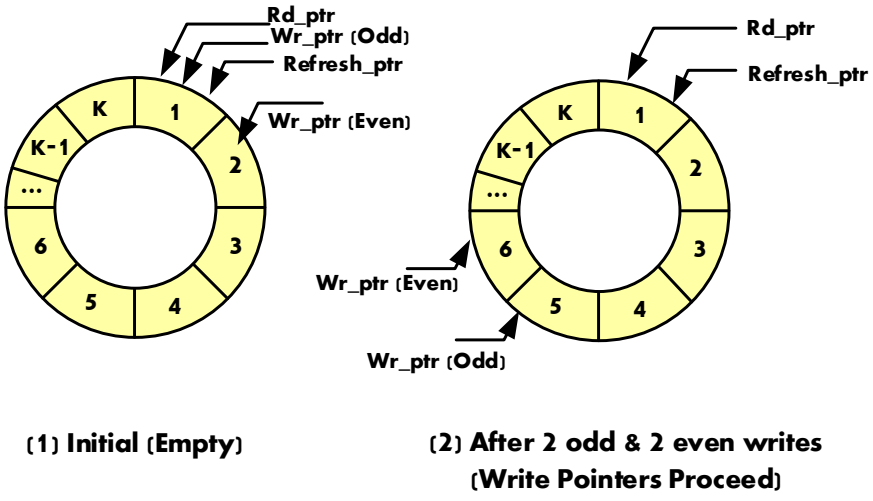

Figure 16: Read / write control logic

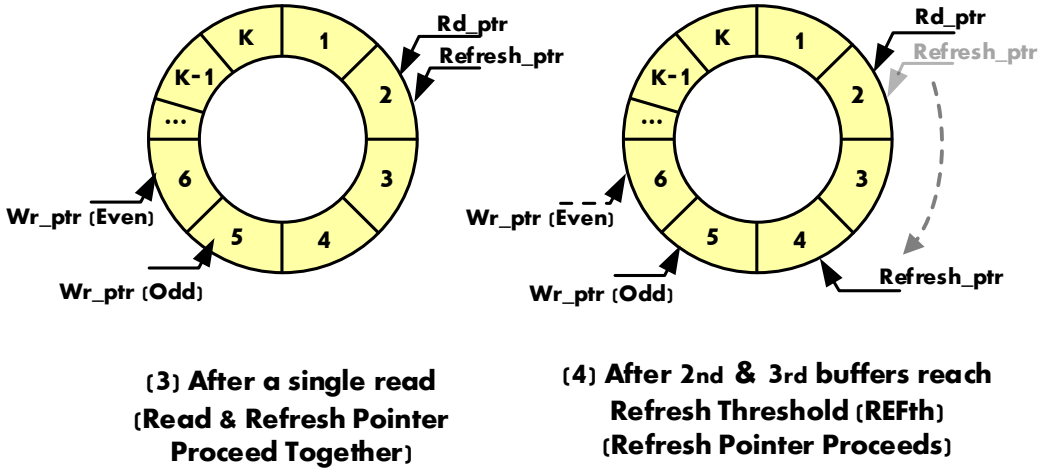

Figure 17: Refresh control logic

The wr_ptr (odd) and wr_ptr (even) correspond to the two banks of the buffer and increment by 2 slots for every update. The control logic works with the bank availability tracker to ensure alternate updates to even and odd bank, so that these 2 pointers are always adjacent. There is only a single read pointer which increments by one for every read. The refresh pointer is set by the refresh logic discussed before. The buffer is

declared empty or full when the read pointer is equal to any of the write pointers after a read or a write. The control logic also bypasses flits if the SA is granted and the VC is empty. This is shown in Figure 17.

The bank availability tracker monitors the write ports and both the write pointers of the VC to decide the destination of the incoming flit.

4.2.4. Bank arbiter for multi-banked buffers

Till now, the retention time of the STT-MRAM was relaxed to get a write latency of 2 cycles. However, this design is scalable for higher write latencies as well. To hide the latency of n cycles (n > 2), n-1 in-channel latches are interfaced with n n:1 MUXes to hide the latency. Figure 18 shows the bank arbiter design for a write latency of 4 cycles. The connections are such that in each cycle, the flit corresponding to Latch L_n drives the output n of the MUX. The $Bank_{sel}$ in such a case will be a 2 bit counter toggling from 0-3 on each I_{clk}.

Latch Jn.	Mux	Mux Input
A	P	0
	Q	1
	R	2
	S	3
B	P	1
	Q	2
	R	3
	S	0
C	P	2
	Q	3
	R	0
	S	1
D	P	3
	Q	0
	R	1
	S	2

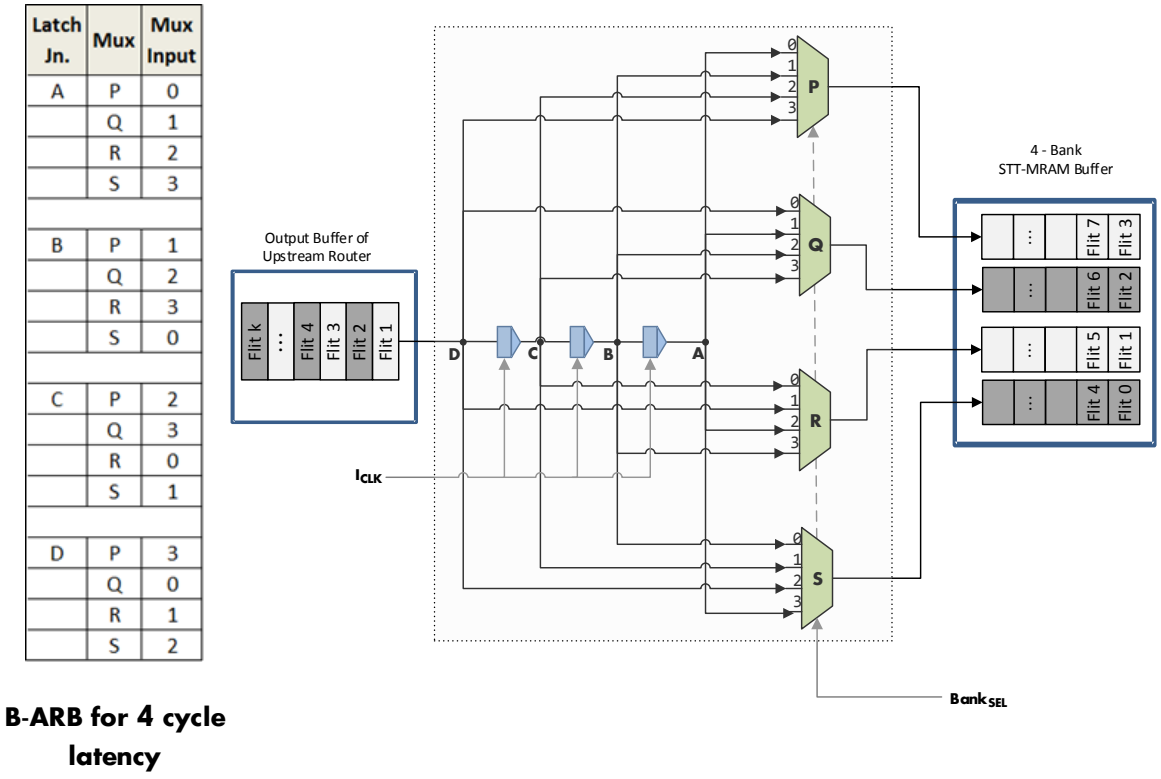

Figure 18: Bank arbiter for 4 cycles

5. PERFORMANCE EVALUATION

5.1. System configuration

We use an in-house cycle accurate NoC simulator to conduct a detailed analysis and performance evaluation of our proposed router and refreshing schemes. The simulator implements a 2 stage speculative router with look ahead routing as discussed in Section 4.1 Each router consists of ‘p’ input ports, ‘v’ virtual channels, a VC arbiter, a crossbar and a switch arbiter. Each VC is essentially a circular FIFO buffer of k-flits, where each flits is 128 bits or 16B in size. We assume 4 virtual channels (‘v’) but vary ‘k’ based on different area budgets and buffer configurations. Routing algorithms like DOR, O1Turn and XY [28] are used with the wormhole switched network.

We use a variety of workloads to realistically evaluate most real-world cases. Benchmarks like Uniform Random (UR), bit complement (BC) and Nearest neighbor (NN) test the effectiveness of our design under synthetic conditions. We use the PARSEC parallel benchmark traces [1] generated via C++ trace reader libraries of netrace [11] to evaluate our design under realistic environments. These trace parsers are incorporated in our simulator.

The CMP configuration parameters that we use are described in Table I. We use a 64 2Ghz processor with alpha ISA for our evaluation. The memory latency is 150 cycles and the flit size is the standard 128 bits. Packet sizes vary with benchmarks, but we also perform analysis by varying packet lengths over synthetic workloads.

Table I: CMP system configuration

System Parameters	Details
Clock frequency	2GHz
# of processors	64, In-order, Alpha ISA
L1 I and D caches	Direct-mapped 32KB (L1I) 4-way 32KB (L1D), 3 cycles
L2 cache	8-way 16MB, 8 cycles 64 banks SNUCA, 256 KB/bank
Cache block size	64B
Coherence protocol	MESI
Memory latency	150 cycles
Flit size	16B
Packet size	1 flit (Benchmark-control) 5 flits (Benchmark-data) 4 flits (Synthetic)

Orion 2.0 [18] is used for area and power analysis for our design. As discussed in Section 3.1, the energy parameters for various retention times for STT-MRAM as compared to SRAM are shown in Table II, and are used to gather energy metrics. The leakage power is expressed in mW per 1 flit buffer. In section 3.1, it was described how the lower area footprint of a 1T-1MTJ cell allows 4x higher packing density as compared to SRAM for the same technology. Throughout the rest of the thesis, the area budgets will be denoted as the respective SRAM budget, denoted by the number of flits per buffer. For eg, SRAM4 means a VC is 4 deep SRAM, STT12 means a 12 deep STT buffer. It should however be noted, that owing to the additional logic described in the previous section, the effective packing density of a STT-MRAM buffer is actually 3.5 times an equivalent SRAM buffer. This is explained in more detail in Section 5.4. Unless otherwise stated, the default write latency of STT-MRAM is chosen to be 2 cycles as discussed in Section 3. However, our design is scalable for higher latencies as well.

Table II: SRAM and STT-MRAM energy parameters

Parameters	SRAM	STT-MRAM		
		10ms	1,000ns	100ns
Read Energy (pJ/flit)	5.25	3.8	3.7	2.7
Write Energy (pJ/flit)	5.25	40.0	22.4	13.7
Leakage Power (mW)	0.028	0.005	0.004	0.003

5.2. Performance analysis with synthetic workloads and benchmark traces

Three configurations with the same area budget were compared for synthetic workloads and PARSEC benchmark traces: SRAM buffer is the baseline, Hybrid buffer [15] and our proposed pure STT-MRAM buffer. We use bit complement (BC), uniform random (UR) and Nearest Neighbor (NN) traffic patterns. For a fair comparison, we choose the Hybrid buffer design which shows the best performance compared to the SRAM baseline. All comparisons are done w.r.t to SRAM4 per VC, i.e. 4 flit deep SRAM buffer per virtual channel. Figure 19 shows the latency vs injection rates for all the traffic patterns. In all cases, the STT-MRAM buffer shows the highest throughput improvement by 19.9%for UR, 23.9%for BC, and 21.2% for NN on average compared to the SRAM buffer, and 5.1% on average compared to the Hybrid buffer across the different traffic patterns. These results indicate that the potential performance degradation due to the multi cycle write latencies of STT-MRAM is offset by both the increased buffer size and the proposed dual bank STT-MRAM scheme, thus resulting in significant performance improvement.

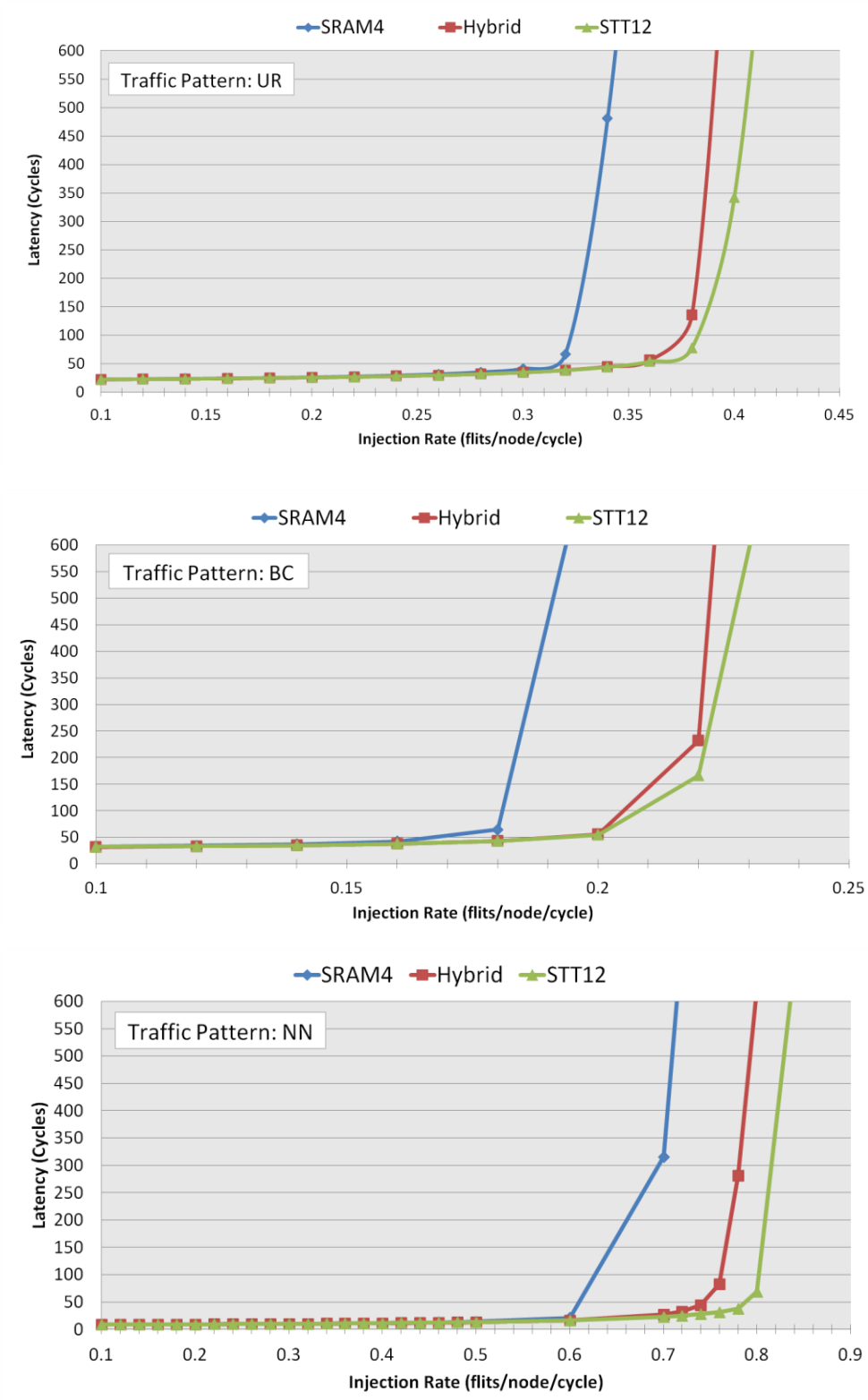

Figure 19: Performance comparison with synthetic workloads

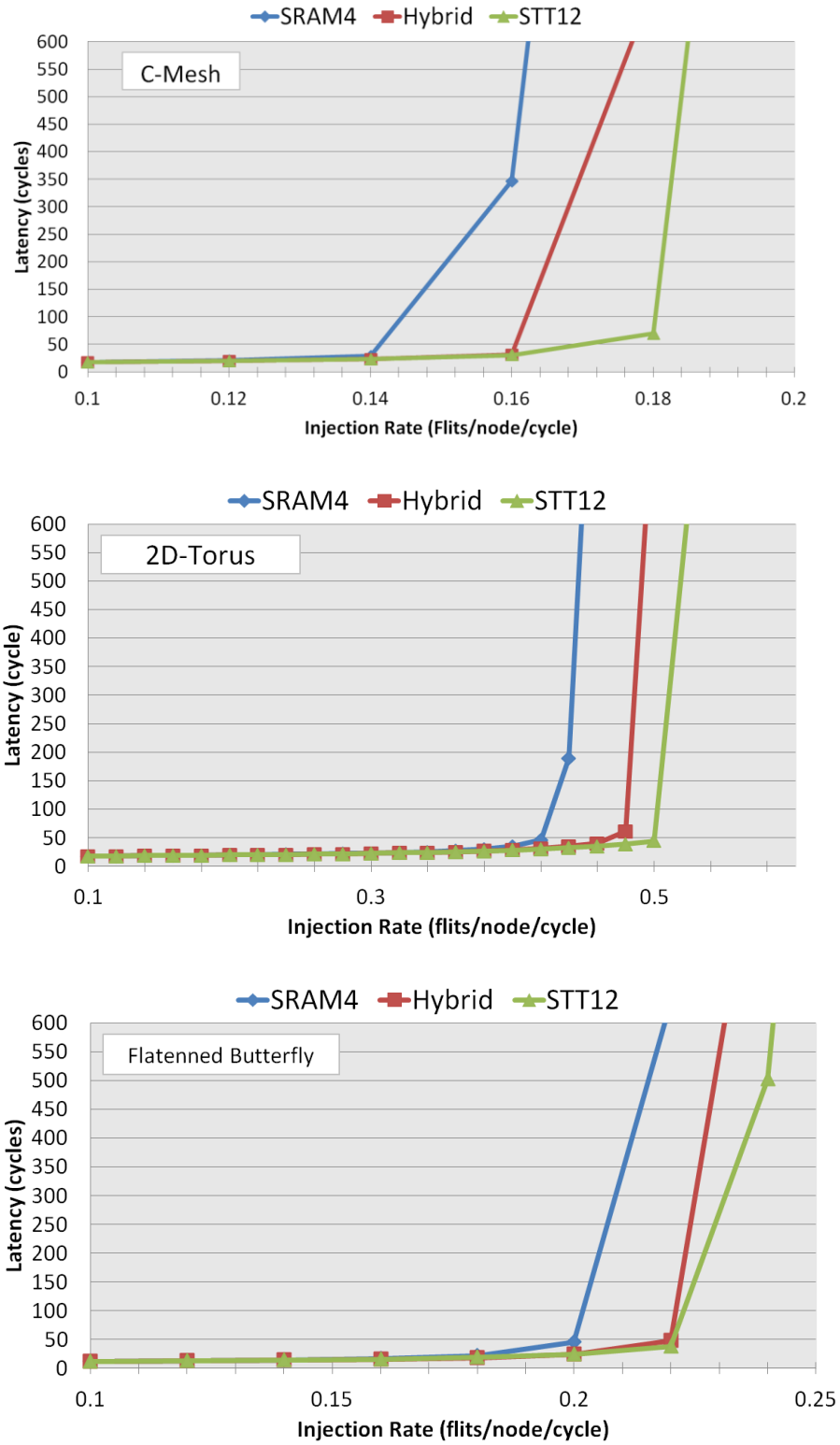

Figure 20: Performance comparison with different topologies

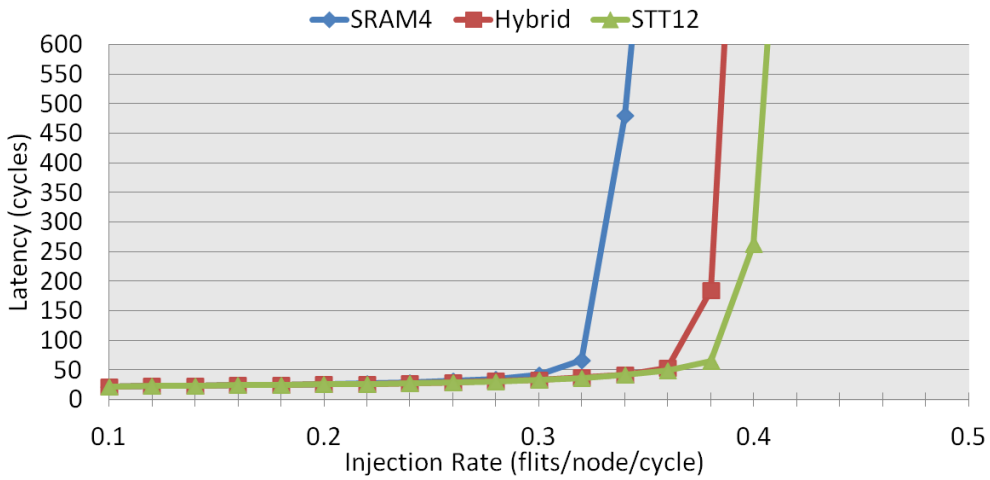

Figure 21: Performance comparison with O1-turn routing

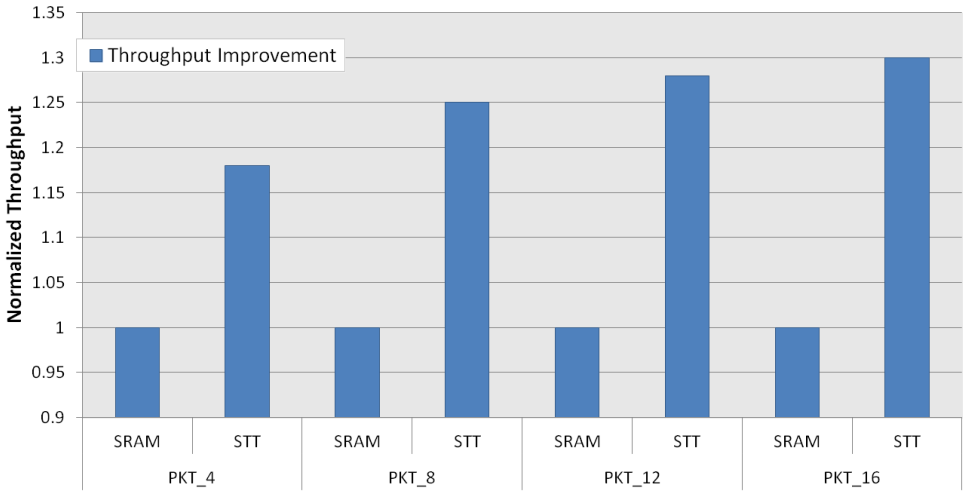

Figure 22: Performance comparison with different packet lengths

We also perform sensitivity studies for other common topologies as well. Figure 20 compares SRAM, Hybrid and STT performance for Concentrated Mesh (C-Mesh), 2D-Torus, and Flattened Butterfly [19] topologies. Throughput improvements of 25.9%, 20.2% and 10% respectively are seen by STT over SRAM, and 5.7%, 9.0%, and 5.2% compared to the Hybrid buffer. To study the impact of routing algorithms on our design,

we compared the schemes with O1 Turn routing in a 8x8 2D mesh. As figure 19 shows, overall throughput increases by 20.5% and 7.9% on average compared to the SRAM and the Hybrid respectively.

Figure 23 shows the average network latency with PARSEC benchmark traces. In general, the STT-MRAM buffer outperforms both the Hybrid and the SRAM buffer, by approximately 6.5% and 15.2% on average, respectively. Specifically, swaptions shows the best performance improvement in the STT-MRAM buffer by 22.1%, and 17.3% compared to the SRAM and the Hybrid buffer, and blackscholes shows the least performance improvement by 12.6%, and 5.8%, respectively. Overall the amount of latency reduction varies depending on the characteristics of each benchmark application.

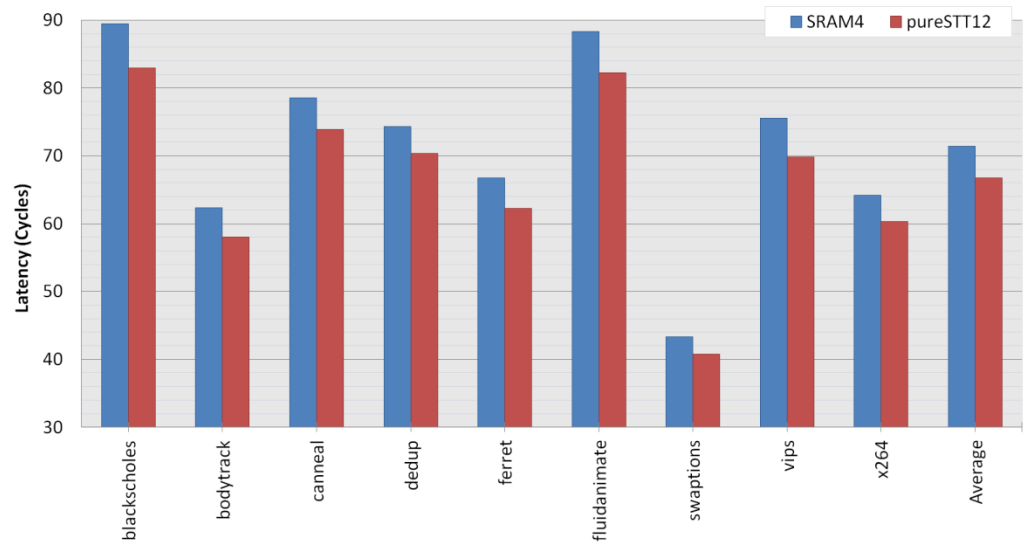

Figure 23: PARSEC performance comparison for SRAM and pure STT

To examine the impact of different packet lengths on network performance in the STT-MRAM buffer scheme, we conduct experiments under the 8x8 2D-mesh with XY

routing algorithm. Figure 22 shows the normalized network throughput improvement under different sizes of a packet: 4, 8, 12, and 16 flits per packet. All results are normalized to that of the baseline SRAM buffer having 4 buffer slots per VC. It clearly shows that the STT-MRAM buffer works better under the long length of packets in terms of throughput improvement. The longest packet, for example, consisting of 16 flits, PKT_16, shows the biggest performance improvement up to 30%in the STT-MRAM buffer over the SRAM baseline. This is because when the buffer capacity is not enough to accommodate a whole unit of packet, the packets in transit tend to spread across multiple nodes, thus impeding subsequent packets from proceeding to their destination, resulting in significant performance degradation.

In Figure 24, we perform a sensitivity analysis to examine the degree of relative performance improvement under the different sizes of the baseline SRAM area budget per VC. Under the five different area budgets: SRAM3, SRAM4, SRAM5, SRAM6, and SRAM7, we compare the throughput improvement of the three different buffer configurations. Under the smallest baseline, SRAM3, the STT-MRAM buffer shows significantly higher throughput improvement by 23%, 11% over the SRAM, and the Hybrid, respectively. However, as the default area budget increases from SRAM3 to SRAM7, the performance gap between the Hybrid and the STT-MRAM decreases. This trend shows that the STT-MRAM buffer is more beneficial as the default area budget in CMP environments becomes tighter.

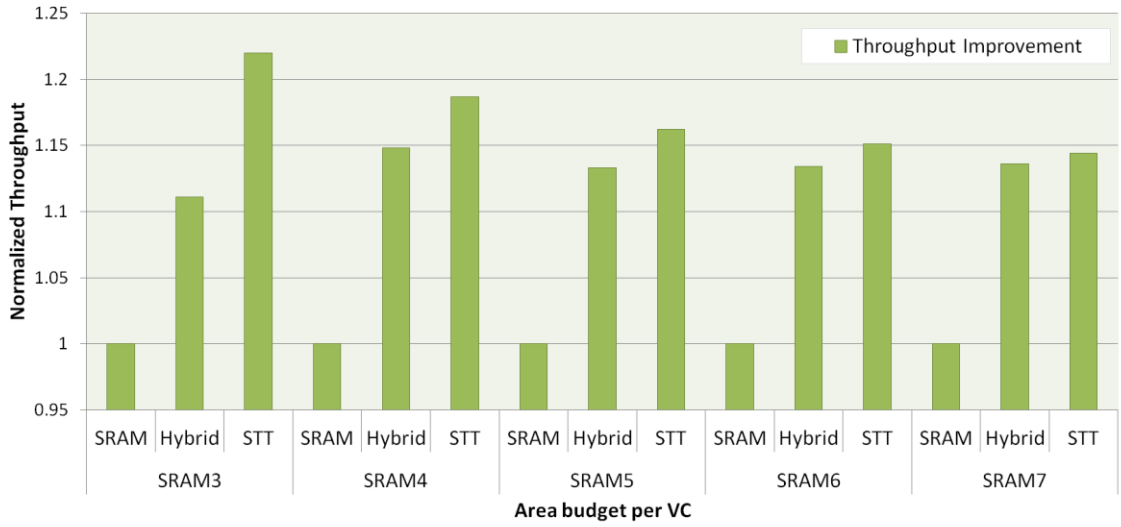

Figure 24: Performance comparison with different area budgets

5.3. Power analysis

Power being a critical design decision in our design, we evaluate the power consumption of the proposed STT-MRAM buffer scheme and compare the effects of the two different refresh schemes described in Section 3.2.2. Figure 26 compares the dynamic buffer power consumption the six different combinations of refresh schemes: Simple, or Global Counter (GC) refresh having 2 different bits of GC (2bit/3bit), and two different SA policies: Round Robin(RR) or Age-based(Age). All results are normalized to that of the SRAM buffer. The age-based SA with 3bit GC consumes the least amount of power across the six different combinations of refresh schemes. Especially compared to the RR with simple refresh, it consumes 39.6% less amount of power on average. On the other hand, the Hybrid scheme consumes 1.7 times more dynamic buffer power, on average compared to the SRAM. This is mainly because of the frequent migration of flits from SRAM to STT-MRAM to hide the long write latency of STT-MRAM. In a network load

below 0.35, both simple and GC refresh scheme consumes less than the half of the power of the baseline SRAM buffer in terms of dynamic buffer power. No flits get dropped in these ranges of injection rates, thus involving no power consumption for the refresh operations. In a high network loads, however, greater than or equal to 0.4, flits get dropped as shown in Figure 25 as the network gets congested, thus triggering refresh operations in the STT-MRAM buffer to maintain the validity of flits. Accordingly, the refresh consumes extra dynamic power, resulting in increasing the overall power consumption of routers.

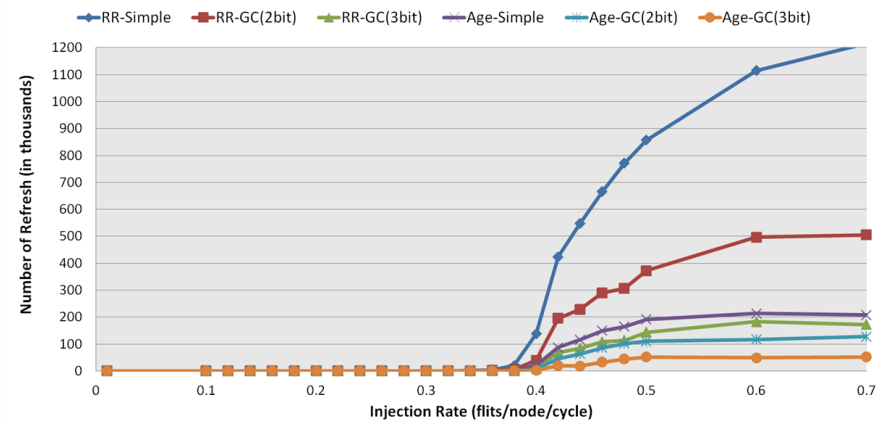

Figure 25: Number of refreshes for different schemes

The total router power consumption for the six different combinations of refresh schemes are shown in Figure 27. The total router power includes not only the leakage and the dynamic power, but also the refresh power required to ensure no flit droppings. As can be seen, for low loads with injection rates < 0.35, an average power saving of 16.5% is seen for pure STT as compared to the SRAM case. Though Table II shows higher dynamic power for STT, near zero leakage energy ultimately dominates these power savings.

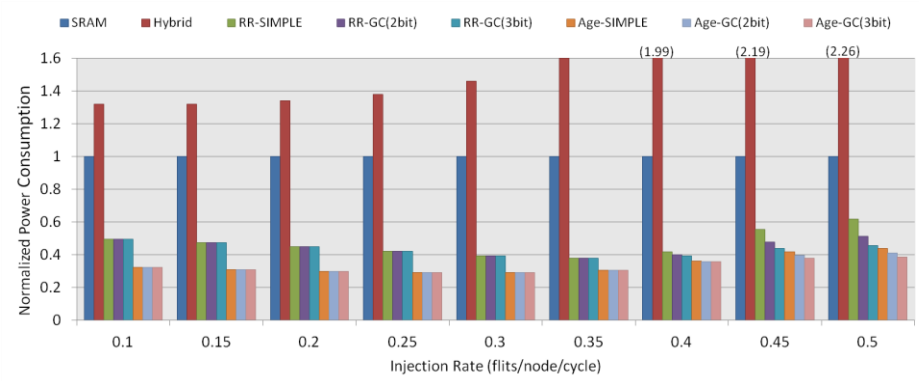

Figure 26: Dynamic power of input buffers for various schemes

As the network gets more congested, the STT-MRAM buffer consumes more power, however, the power consumption of the 3bit GC with age-based SA is still much less than the SRAM buffer by 18.2% on average across the different injection rates. The Hybrid scheme consumes much larger power than any STT-MRAM scheme, and consumes 12% more power at a maximum injection rate i.e. 0.5/flits/node/cycle because of the same reason mentioned above: the frequent migration operations.

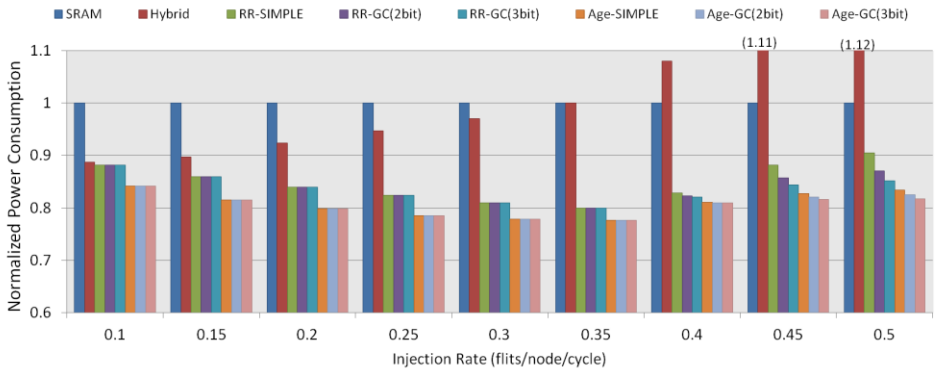

Figure 27: Total router power for various schemes

5.4. Area analysis

For area modeling, we use ORION 2.0 [18] which provides value estimates of the router area using gate level netlists to model the basic NoC router elements e.g. FIFO buffers, crossbar and switch arbiters. Basically, ORION 2.0 adds an additional 10% to the total area to account for global white space. Table III provides the area estimates for the different configurations of SRAM, Hybrid and STT-MRAM, respectively, under the same SRAM area budget in 32nm technology.

Table III: Router area (in μm^2) for different area budgets in 32nm technology

Default Area Budget	Configuration	# of Buffer Slots	BUFFER AREA		VC Allocator	SW Arbitor	Xbar	Extra Logic (PureSTT)	Total Router Area (μm^2)
			SRAM	STT-MRAM					
SRAM4	SRAM4	4	62535.05	-	8680.51	868.05	192087.4	-	264170.98
	HYBRID (SRAM3_STT4)	7	49535.63	10684.45				-	261856.01
	STT12	12	-	47662.69				10844.7	260143.31
SRAM5	SRAM5	5	75534.48	-	(same as above)	(same as above)	(same as above)	-	277170.4
	HYBRID (SRAM3_STT8)	11	49535.63	21368.91				-	272540.47
	STT15	15	-	60920.49				10844.7	273401.11
SRAM6	SRAM6	6	88533.9	-	(same as above)	(same as above)	(same as above)	-	290169.83
	HYBRID (SRAM3_STT12)	15	49535.63	32053.37				-	283224.93
	STT18	18	-	74178.29				10844.7	286658.92

The lengths of bit-line and word-line decide the area of a single buffer slot and are dependent on the number of read and write ports. STT-MRAM cells with reduced retention times are modeled with slightly scaled up transistors used in [17], which gives us an area ratio of $31.2F^2/146F^2$ with respect to the SRAM Cell. It is noted that the 2 stage switch arbiter, VC allocator and crossbar area are unchanged across configurations,

but the difference comes from the area of the buffers and the extra logics hiding the multi-cycles write latency of STT-MRAM. The extra logic area is mainly taken up by flit wide MUXes, an additional VC decoder per port. The overheads of the multi-port and additional combinatorial logics reduce the effective density from 4 STT-MRAM cells per SRAM cell to 3, and thus a STT18 configuration corresponds to a SRAM6 area. In case of the Hybrid scheme, buffers are single-port and have relatively negligible extra logic area, thus utilizing the full STT-MRAM density of four times the SRAM.

6. CONCLUSION

In this thesis, a new NoC router input buffer design is proposed, that utilizes only STT-MRAM cells as its constituents, providing higher network throughputs while saving overall power consumption for interconnection networks. Advantages of high density and low leakage are major attractions of using STT, while low write performance is mitigated by relaxing its volatility. The multi-cycle write latency is hidden using a multi-banked buffer design with simple link buffers inside the downstream router. Cost-efficient refresh schemes are proposed that prevent flit dropping that are a consequence of high performance low volatility of STT-MRAM. Simulation results indicate that by reducing the retention time to100ns, a performance improvement of 21.3% and 15.2% on average is seen for synthetic and PARSEC benchmarks respectively, as compared to the SRAM based input buffers under same router area.

REFERENCES

[1] C. Bienia, S. Kumar, J. P. Singh, and K. Li, "The PARSEC Benchmark Suite: Characterization and Architectural Implications," in Proceedings of PACT, 2008.

[2] Y. Cao, T. Sato, M. Orshansky, D. Sylvester, and C. Hu, "New Paradigm of Predictive MOSFET and Interconnect Modeling for Early Circuit Simulation," in Proceedings of IEEE Custom Integrated Circuits Conference, 2000.

[3] M. T. Chang, P. Rosenfeld, S.L. Lu, and B. Jacob, "Technology Comparison for Large Last-Level Caches: Low-Leakage SRAM, Low Write-Energy STT-RAM, and Refresh-Optimized eDRAM," in Proceedings of HPCA, 2013.

[4] X. Chen and L.S. Peh, "Leakage Power Modeling and Optimization in Interconnection Networks," in Proceedings of ISLPED, 2003.

[5] W. J. Dally, "Virtual-Channel Flow Control," IEEE Trans. Parallel Distrib. Syst., vol. 3, pp. 194–205, March 1992.

[6] W. J. Dally and C. L. Seitz, "Deadlock-Free Message Routing in Multiprocessor Interconnection Networks," IEEE Trans. Comput., vol. 36, pp. 547–553, May 1987.

[7] D.Bedau, H.Liu, J. J.Bouzaglou, A.D.Kent, J.Z.Sun, J.A.Katine, E.E.Fullerton, and S.Mangin, "Ultrafast Spin-Transfer Switching in Spin Valve Nanopillars with Perpendicular Anisotropy," Applied Physics Letters, vol. 96, p. 022514, 2010.

[8] Z. Diao, Z. Li, S. Wang, and Y. Ding, "Spin-Transfer Torque Switching in Magnetic Tunnel Junctions and Spin-Transfer Torque Random Access Memory," Journal of Physics: Condensed Matter, vol. 19, p. 165209, 2007.

[9] M. Galles, "Scalable Pipelined Interconnect for Distributed Endpoint Routing: The SGI SPIDER Chip," in Proceedings of Hot Interconnect 4, 2009.

[10] X. Guo, E. Ipek, and T. Soyata, “Resistive Computation: Avoiding the Power Wall with Low-Leakage, STT-MRAM Based Computing,” in Proceedings of ISCA, 2010.

[11] J. Hestness, B. Grot, and S. W. Keckler, “Netrace: Dependency- Driven Trace-Based Network-on-Chip Simulation,” in Proceedings of NoCArc, 2010.

[12] Y. Ho, G. M. Huang, and P. Li, “Nonvolatile Memristor Memory: Device Characteristics and Design Implications,” in Proceedings of ICCAD, 2009.

[13] Y. Hoskote, S. Vangal, B. N. Singh, Avind, and S. Borkar, “A 5-GHz Mesh Interconnect for a Teraflops Processor,” in Proceedings of MICRO, 2007.

[14] ITRS, “International Technology Roadmap for Semiconductors: 2009 Executive Summary,” http://www.itrs.net/Links/2009ITRS/Home2009.htm.

[15] H. Jang, B. S. An, N. Kulkarni, K. H. Yum, and E. J. Kim, “A Hybrid Buffer Design with STT-MRAM for On-Chip Interconnects,” in Proceedings of NOCS, 2012.

[16] L. Jiang, B. Zhao, Y. Zhang, J. Yang, and B. R. Childers, “Improving Write Operations in MLC Phase Change Memory,” in Proceedings of HPCA, 2012.

[17] A. Jog, A. K. Mishra, C. Xu, Y. Xie, N. Vijaykrishnan, R. Iyer, and C. R. Das, “Cache Revive: Architecting Volatile STT-RAM Caches for Enhanced Performance in CMPs,” in Proceedings of DAC, 2012.

[18] A. B. Kahng, B. Li, L.-S. Peh, and K. Samadi, “ORION 2.0: A Fast and Accurate NoC Power and Area Model for Early-Stage Design Space Exploration,” in Proceedings of DATE, 2009.

[19] J. Kim, J. Balfour, and W. Dally, “Flattened Butterfly Topology for On-Chip Networks,” in Proceedings of MICRO, 2007.

[20] A. K. Kodi, A. Sarathy, and A. Louri, “iDEAL: Inter-Router Dual- Function Energy and Area-Efficient Links for Network-on-Chip (NoC) Architectures,” in Proceedings of ISCA, 2008.

[21] S. Ma, N. E. Jerger, and Z. Wang, "Whole Packet Forwarding: Efficient Design of Fully Adaptive Routing Algorithms for Networks-on- Chip," in Proceedings of HPCA, 2012.

[22] A. K. Mishra, X. Dong, G. Sun, Y. Xie, N. Vijaykrishnan, and C. R. Das, "Architecting On-Chip Interconnects for Stacked 3D STT-RAM Caches in CMPs," in Proceedings of ISCA, 2011.

[23] C. A. Nicopoulos, D. Park, J. Kim, N. Vijaykrishnan, M. S. Yousif, and C. R. Das, "ViChaR: A Dynamic Virtual Channel Regulator for Network-on-Chip Routers," in Proceedings of MICRO, 2006.

[24] J. D. Owens, W. J. Dally, R. Ho, D. Jayasimha, S. W. Keckler, and L. S. Peh, "Research Challenges for On-Chip Interconnection Networks," in Proceedings of MICRO, 2007.

[25] L. S. Peh and W. J. Dally, "A Delay Model and Speculative Architecture for Pipelined Routers," in Proceedings of HPCA, 2001.

[26] A. Raychowdhury, D. Somasekhar, and T. Karnik, "Design Space and Scalability Exploration of 1T-1STT MTJ Memory Arrays in the Presence of Variability and Disturbances," in Proceedings of IEDM, 2009.

[27] N. D. Rizzo, M. DeHerrera, J. Janesky, B. Engel, J. Slaughter, and S. Tehrani, "Thermally Activated Magnetization Reversal in Submicron Magnetic Tunnel Junctions for Magneto-resistive Random Access Memory," Applied Physics Letters, vol. 80, p. 2335, 2002.

[28] D. Seo, A. Ali, W. T. Lim, N. Rafique, and M. Thottethodi, "Near- Optimal Worst-Case Throughput Routing for Two-Dimensional Mesh Networks," in Proceedings of ISCA, 2005.

[29] C.W.Smullen, V. Mohan, A. Nigam, S.Gurumurthi, and M.R. Stan, "Relaxing Non-Volatility for Fast and Energy-Efficient STT-RAM Caches," in Proceedings of HPCA, 2011.

[30] G. Sun, X. Dong, Y. Xie, J. Li, and Y. Chen, "A Novel Architecture of the 3D Stacked MRAM L2 Cache for CMPs," in Proceedings of HPCA, 2009.

[31] Z. Sun, X. Bi, H. H. Li, W. F. Wong, Z. L. Ong, X. Zhu, and W. Wu, "Multi Retention Level STT-RAM Cache Designs with a Dynamic Refresh Scheme," in Proceedings of MICRO, 2011.

[32] T.Dunn and A.Kamenev, "Optimization of the Current Pulse for Spin- Torque Switches," Applied Physics Letters, vol. 98, p. 143109, 2011.

[33] H. Wang, L. S. Peh, and S. Malik, "Power-driven Design of Router Microarchitectures in On-chip Networks," in Proceedings of MICRO, 2003.

[34] X. Wu, J. Li, L. Zhang, E. Speight, R. Rajamony, and Y. Xie, "Hybrid Cache Architecture with Disparate Memory Technologies," in Proceedings of ISCA, 2009.

[35] A. V. Yakovlev, A. M. Koelmans, and L. Lavagno, "High-Level Modeling and Design of Asynchronous Interface Logic," IEEE Design and Test of Computers, vol. 12, pp. 32–40, 1995.

[36] T. T. Ye, L. Benini, and G. D. Micheli, "Analysis of Power Consumption on Switch Fabrics in Network Routers," in Proceedings of DAC, 2002.

[37] Y.Huai, "Spin-Transfer Torque MRAM (STT-MRAM): Challenges and Prospects," AAPPS Bulletin, vol. 18, pp. 33–40, 2008.

[38] H. Zhao, A. Lyle, Y. Zhang, and P. K. Amiri, "Low Writing Energy and Sub Nanosecond Spin Torque Transfer Switching of In-Plane Magnetic Tunnel Junction for Spin Torque Transfer Random Access Memory," Journal of Applied Physics, vol. 109, pp. 07C720–3, 2011. 12