A Multibank Buffer Design with STT-MRAM
for High-Bandwidth Low-Power On-Chip Interconnects

Hyunjun Jang, Rahul Boyapati, Rohan Kansal, Ki Hwan Yum and Eun Jung Kim
Department of Computer Science and Engineering
Texas A&M University
College Station, Texas, 77843-3112
Email: {hyunjun, rahulboyapati, rkansal, yum, ejkim}@cse.tamu.edu

Abstract

Network-on-Chip (NoC) is a widely accepted inter-core communication infrastructure for Chip Multiprocessors (CMPs). It is critical that NoC provides both low latency and high bandwidth within limited on-chip power and area budgets. Spin-Transfer Torque Magnetic RAM (STT-MRAM), a high density and low leakage memory, offers more buffer capacities with the same die footprint, thus helping increase network throughput in NoC routers. However, its long latency and power overhead associated with write operations still need to be addressed. In this paper, we propose the first NoC router design that uses only STT-MRAM, which provides much larger buffer space with less power consumption. To hide the multicycle writes, we employ a multibank STT-MRAM buffer, a virtual channel with multiple banks where every incoming flit is seamlessly pipelined to each bank alternately. Our STT-MRAM design has aggressively reduced the retention time, resulting in a significant reduction of the latency and power overheads of write operations. To overcome flit losses caused by the reduced retention time, we propose cost-efficient dynamic buffer refresh schemes that minimize unnecessary refreshes with minimum hardware overheads. Simulation results show that the proposed STT-MRAM NoC router improves throughput by 21.6% and achieves 17% savings in the total router power compared with a conventional SRAM based NoC router.

1. Introduction

Switch-based Network-on-Chip (NoC) has become a popular architecture orchestrating chip-wide communication in Chip Multiprocessors (CMPs). NoC should be carefully designed due to its inherent constraints of the restricted power and area budgets in a chip. NoC consumes up to 28% of the chip power [25], and among the different components comprising on-chip interconnects, buffers are the largest leakage power consumers in NoC routers, consuming about 68% of the total router leakage power [12]. Buffers also consume significant dynamic power [51, 56], and this consumption increases rapidly as data flow rates increase [56]. Consequently, designing an innovative buffer structure plays a crucial role in architecting high performance and low power on-chip interconnects.

Spin-Transfer Torque Magnetic RAM (STT-MRAM) [38, 44, 48, 57] is a promising next generation memory technology that can replace conventional RAMs due to its near-zero leakage power and high density. Adopting STT-MRAM in NoC has significant merits since an on-chip router can provide larger input buffers under the same area budget compared with conventional SRAM routers. Larger input buffers contribute to improving throughput, which results in enhanced system performance with less power consumption. STT-MRAM is CMOS-compatible, and provides virtually infinite write endurance [20] compared with other memory technologies such as Phase Change Memory (PCM) [28], Flash, and Memristor [24], which makes STT-MRAM a more viable solution as an on-chip memory that should tolerate frequent write accesses. Besides, STT-MRAM is immune to the radiation induced soft errors [46, 47], thus providing robust cell storages, and can scale beyond 10 nm technology [15]. However, the weaknesses of STT-MRAM, long latency and high power consumption in write operations, should be properly addressed because fast accesses to on-chip memories must be assured for high performance NoCs.

Several studies have been performed to address these write speed and energy limitations of STT-MRAM in designing caches and NoC routers. An adaptive block placement and migration policy for hybrid STT-RAM and SRAM last level caches has been proposed in [52]. An SRAM/MRAM hybrid cache with 3D stacking structure was proposed in [45]. A region-based hybrid cache [54] with small fast SRAM and large slow MRAM mitigates performance degradation and energy overheads. For NoC routers, an SRAM/STT-MRAM hybrid buffer [27] shows substantial throughput improvements across various workloads. However, the inevitable use of SRAM to hide the multicycle writes of STT-MRAM sacrifices area, and wastes significant dynamic power in migrating data between the disparate memories. The leakage power overhead due to SRAM also increases as network scale grows and technology scales down [5].

In this paper, we propose the first NoC router design that uses only STT-MRAM in buffers. By eliminating SRAM, it offers much larger buffer space with less power consumptions. To hide the multicycle write latencies of STT-MRAM, we propose a novel pipelined input buffer design, a multibank STT-MRAM buffer, which is a virtual channel (VC) with multiple banks where every incoming flit is delivered to each bank alternately via a simple latch inside a router. Through this, we can avoid performance degradation while consuming less area and power.

For our design, we use the write latency reduction technique [48], which sacrifices the data retention time of an Magnetic Tunnel Junction (MTJ), a bit storage of STT-MRAM. This can be possible due to the short intra-router latency of a flit in on-chip routers as shown in Figure 1, where the average intra-router latency is less than 3 cycles. However, for applications that exhibit bursty communication and have heavy loads, we observe that flits are staying in STT-MRAM buffers longer than

1 An intra-router latency is the time interval between the arrival of a flit at an input buffer and the departure from a router through a crossbar.
2 See Section 5 for detailed system configuration.
a given retention time, which results in flit losses\(^3\). This is because some flits have fairly high intra-router latencies while most of the flits are clustered around low intra-router latencies less than 10 cycles as shown in Figure 2. These lost flits incur noticeable performance losses especially when the flits are parts of control packets carrying critical cache coherence information. On average, 78.7\% of traffic is such single-flit control packets in PARSEC benchmarks [36]. Therefore, we propose cost-efficient dynamic buffer refresh schemes, the processes in which cells’ values are kept valid by triggering refreshes in a timely manner with minimum hardware overheads.

The main contributions of this paper are as follows:

- We present a detailed analysis on design tradeoffs of an MTJ especially in terms of write performance, write power, and retention time, which are suitable for performance- and power-efficient NoCs.
- We propose a novel multibank input buffer design, which is implemented entirely with STT-MRAM and delivers optimal power saving and performance improvement.
- We suggest cost-efficient dynamic buffer refresh schemes: a simple refresh scheme and a global counter refresh scheme, which selectively trigger buffer refreshments in a power efficient manner to maintain the validity of flits.
- We achieve 21.6\% throughput improvement and 17% total power saving compared to a conventional SRAM based router with the proposed STT-MRAM router scheme.

2. Background

In this section, we review key features of STT-MRAM and analyze design tradeoffs of an MTJ cell in terms of switching time (the time taken for completing a write operation in an MTJ cell, namely write latency), switching current (the power required to change an MTJ cell value, namely write power), and data retention time.

2.1. STT-MRAM

STT-MRAM is a next generation memory technology that exploits magnetoresistance for storing data [38, 44, 48, 57]. In STT-MRAM, each data bit is stored in an MTJ, a fundamental building block. An MTJ consists of three layers: two ferromagnetic layers and a Magnesium Oxide (MgO) tunnel barrier layer in the middle as shown in Figure 3. The magnetization direction of the bottom ferromagnetic layer is fixed. The spin of the electrons in the top layer is influenced by a high-amplitude current pulse that propagates through the fixed layer. Depending on the current, the spin polarity of the free layer changes to either parallel or anti-parallel to that of the fixed layer. The parallel indicates a zero state and the anti-parallel a one state as shown in Figure 3. A single MTJ module is coupled with an NMOS transistor to form a basic memory cell of STT-MRAM, called a 1T-1MTJ.

2.2. STT-MRAM Design Considerations

- Retention Time: The nonvolatility of an MTJ is quantitatively measured by the data retention time, which is the maximum time duration for which stored data can remain in an MTJ [29, 44]. The data retention time, \(T_{\text{ret}}\), of an MTJ is defined as follows [42].

\[
T_{\text{ret}} = 1\text{ns} \cdot e^\Delta
\]

\(\Delta\) is the thermal factor that estimates thermal stability of an MTJ, and it is proportional to the saturation magnetization \(M_s\), the in-plane anisotropy field \(H_k\), and the volume of an MTJ cell \(V\) as follows [11]. \(T\) denotes the working temperature.

\[
\Delta \propto \frac{M_s H_k V}{T}
\]

We decrease the thermal factor by reducing \(M_s\) and \(H_k\), leading to reduced retention-time STT-MRAM [48]. Note that the decreased thermal stability might hurt the soft error robustness of STT-MRAM, thus potentially incurring random bit-flips [43], but this problem can be solved by adopting suitable soft error protection schemes for STT-MRAM [47, 49].

- Switching Current and Switching Time: In a precessional switching mode [41] where an MTJ switching time \(T_s\) is short (<3 ns), the required current density, \(J_c(T_s)\), is determined as follows.

\[
J_c(T_s) = J_{c0} + \frac{C}{T_s}
\]
where \( J_{0} \) is a switching threshold current density that also depends on \( M_s \) and \( H_K \) [57] like the thermal factor \( \Delta \). \( C \) is a constant affected by the initial angle between the magnetization vector of the free layer and the easy axis [48]. Reducing the retention time causes the thermal factor to decrease, which reduces \( M_s \) and \( H_K \), and eventually decreases \( J_{0} \). Therefore, with smaller \( J_{0} \), we can achieve a shorter switching time with the reduced current density, \( J_s (T_s) \).

Figure 4 depicts the inverse relationship between the switching current \( (J_s(T_s)) \) and the switching time \( (T_s) \) under four different MTJ retention times ranging from 10 years to 100 ns. The retention time curves in Figure 4 are plotted based on previous studies [29, 44, 48], where the retention time is reduced up to tens of \( \mu \)s level, and for our STT-MRAM buffer design, we further reduce the retention time to 100 ns (proven to be feasible in [10]) based on MTJ device equations [29] and simulation with the PTM model [4] under 32 nm technology. As we further reduce the retention time, the required MTJ switching time and switching current get decreased accordingly, leading to better write performance and less write power overhead. When fixing the switching time at 1.0 ns, for instance, we can reduce the write current by 45.2% by relaxing the retention time from 10 ms to 100 ns. Based on this analysis, we integrate the buffer-level SRAM and STT-MRAM models in NVsim [14] and simulation results are shown in Table 2.

- **Cell Area**: As an area model of STT-MRAM, we refer to ITRS projections [26] as well as the model used in [20], where a 1T-1MTJ size is 30 \( F^2 \). When we assume that an SRAM cell size is approximately 146 \( F^2 \) under 32 nm technology, one SRAM cell can be substituted by at least four STT-MRAM cells under the same area budget. An STT-MRAM cell area is mostly determined by the NMOS transistor size since the MTJ cell is much smaller than the transistor [20].

- **Impact of Process Technology**: Applying different process technologies can affect the overall STT-MRAM power-performance cost. As process technologies scale down, the future STT-MRAM is predicted to achieve a significantly smaller cell size, faster read/write with lower power consumption while maintaining the non-volatility property [15, 49]. For advanced technologies such as 22 nm, NVsim [14] circuit-level simulation shows that the cell area is decreased by 48.4\%, the read/write dynamic power by 13\%, and the leakage power by 41.4\% compared to those of 32 nm. The write delay can also be decreased further due to the smaller cell size and less current required for bit-flips. These trends indicate that STT-MRAM will become a more viable option for cost-efficient NoC routers.

Figure 5: Performance Comparison between SRAM and STT-MRAM based NoC Routers under the Same Area Budget

3. **Motivation**

In this section, we present key motivations that drive us to STT-MRAM based NoC routers for power and performance co-optimization.

3.1. **STT-MRAM for NoC Routers**

Figure 5 compares the performance of an NoC router equipped with SRAM, STT-MRAM, and ideal STT-MRAM buffers having no write delays. Under the same area budget, STT-MRAM provides 4 times more buffer space as described in Section 2.2. Due to the long write delay of STT-MRAM, \( STT16 \) (baseline), the SRAM based router shows far better performance, but once we completely hide the write delay of STT-MRAM, \( STT16 \) (no-lat), the overall throughput is increased by 20\% compared with that of SRAM with no zero-load penalty. Also, STT-MRAM has near-zero leakage power, thus consuming much less total power compared with SRAM as shown in Section 5.3. SRAM appears to be much more power hungry than STT-MRAM, and consequently gives STT-MRAM performance leeway in a power constrained NoCs. This motivates us to adopt STT-MRAM for NoC routers for better performance and less power consumption.

3.2. **Determining Proper Retention and Switching Times**

Based on Figure 4, for power- and performance-efficient NoC routers, it is important to identify what the ideal/feasible retention time should be. This is because significant retention time reduction will make the STT-MRAM buffer highly volatile, leading to performance degradation due to dropped flits, while increasing the retention time will negatively affect write performance and energy. Considering these tradeoffs, to decide a suitable range of the retention time for the STT-MRAM buffer, we measure the average intra-router latency of target applications because it is the main factor affecting flits’ lifetime. If flits stay in the STT-MRAM buffer longer than a given retention time, they get invalidated. We conduct experiments with traces from PARSEC benchmarks [3], where all results are measured under the same area budget, 6 SRAM slots per VC, for input buffers. Figure 1 shows the result, and based on these short intra-router latencies, it is reasonable to further reduce the retention time to \( nanoseconds \), rather than \( milliseconds \) or \( microseconds \) which is widely used in designing caches with STT-MRAM [29, 44, 48], thus leading to the least write and energy overheads among four different retention times in Figure 4.

In an ideal case, STT-MRAM write latency should be equal to that of SRAM, thus writing to STT-MRAM must be done in a single cycle, which corresponds to less than 0.5 ns in 2
GHz clock frequency. Such fast write times of less than 0.5 ns have proved possible [10, 49], but as shown in Figure 4, it requires rather strong currents, and is far from the optimal efficiency [50]. Even with the shortest retention time, therefore, we conclude that it is inevitable to have more than 1-cycle latency for a write operation in the STT-MRAM buffer.

Our proposed scheme exploits these observations to accelerate STT-MRAM write speeds with less power consumption. In Section 4.2, we propose router architectures that effectively hide the multicycle write latencies of STT-MRAM.

In summary, for power-performance co-optimized STT-MRAM buffer design, as detailed in Section 2.2, we reduce the retention time to 100 ns, and through this, 2-cycle write latency, corresponding to 1 ns in 2 GHz clock frequency, is achieved with 71.35 µA of switching current (See the point where 100 ns retention and 1.0 ns switching time intersect in Figure 4) with 30 $F^2$ of STT-MRAM cell size.

3.3. Avoiding Flit Losses

A key challenge incurred by the retention time reduction is to overcome the potential flit losses occurring when flits oversstay a given retention time in STT-MRAM buffers. We observe three main NoC factors (Applications, Retention Times, and Network Scales) affecting the validity of flits in an STT-MRAM router. As shown in Figure 6(a), under the retention time of 100 ns, overstayed flits get dropped. Each benchmark application has different traffic patterns over its lifetime, thus showing different number of flits dropped. In particular, canneal, fluidanimate, and dedup show higher flit dropping rates than others due to their bursty traffic characteristics. Figure 6(b) shows that, under uniform random (UR) traffic, as the retention time of STT-MRAM gets decreased, maintaining the validity of flits in the STT-MRAM buffer gets difficult, especially when networks become congested. Increased network scale also affects the number of flits dropped due to increased network contention as shown in Figure 6(c). At the same injection rate, under UR, 3.3 times more flits get dropped in a (16x16) mesh, on average, compared with those of an (8x8) mesh. From these observations, hence, it is imperative to take appropriate actions applicable to NoCs to prevent such flit losses.

There are several studies addressing these data losses in STT-MRAM through various refresh techniques [29, 44, 48], but their application domains are only limited to caches, which are totally different from buffers in an NoC router. Caches have relatively large capacities and require longer data retention times compared with small-sized FIFO input buffers. Multiple retransmission schemes [16, 22] minimizing buffering requirements have been proposed for bufferless routing. They cannot be directly applicable, since unlike bufferless routers, our work requires much larger outside buffering (for injection and reassembly) to hold all the packets in-fly in the network. This is because the number of packets in-fly are significantly greater in our scheme due to the larger buffer space available in the network by virtue of STT-MRAM. In Section 4.3, we propose cost-efficient dynamic buffer refresh schemes eliminating flit dropping with less refresh overheads.

4. STT-MRAM Router Architecture

In this section, we describe a baseline router architecture with its buffer structure and present an STT-MRAM based router in detail. The key design goal of the proposed scheme is to enable flits to be written into buffers with no additional time delay.

4.1. Baseline Router Architecture

The baseline NoC router architecture is depicted in Figure 7, which is similar to that used in [34] employing several features for latency reduction, including speculation [53] and lookahead routing [17]. Each arriving flit goes through 2 pipeline stages in the router: routing computation (RC), VC allocation (VA), and switch arbitration (SA) during the first cycle, and switch traversal (ST) during the second cycle. The lookahead routing generates routing information of the downstream router for an incoming flit prior to the buffer write, thus removing the RC stage from the critical path. Each router has multiple VCs per input port and uses flit-based wormhole switching [9]. Credit-based VC flow control [8] is adopted to provide the back-pressure from downstream to upstream routers. The necessity for ultra-low latency leads to a parallel FIFO buffer shown in Figure 10(a). Unlike a serial FIFO implementation, the parallel structure eliminates unnecessary intermediate processes making a flit traverse all buffer entries until it leaves the buffer [55]. The read and write pointers in the parallel FIFO regulate the operations of the input and output MUXes, and the two pointers are controlled by a VC control logic, which generates proper
read and write pointer values and status signals.

4.2. STT-MRAM Router Design

For conventional SRAM buffers, incoming flits are written to their designated buffers with no delay due to the short SRAM write latency. On the contrary, when we replace SRAM with STT-MRAM, only a single flit can be written to a buffer every \( n \) cycles, which causes subsequent incoming flits to be delayed. To guarantee seamless traversal of flits across the network, we propose a multibank STT-MRAM buffer that hides the long write latency inherent in STT-MRAM.

4.2.1. Multibank STT-MRAM Buffer

The multibank buffer scheme can be used to hide \( n \)-cycle write latency of STT-MRAM. For example, to hide 2-cycle write delay of STT-MRAM buffer, we divide each VC into two banks where every incoming flit is seamlessly pipelined to each bank alternately via a simple latch inside a router. Note that many studies \([12, 13, 31, 18]\) explore the latch-based NoC pipeline design, where latches along the link are utilized as temporary buffers that can hold and release data when necessary. The simple latch in this paper is controlled by a control block \([13]\) interfaced with the NoC clock, having the dual function of switching between storing and transmitting data. Let us refer to the two banks as Odd and Even banks, respectively, and incoming flits from upstream routers as Odd and Even flits as shown in Figure 8(a). Every odd numbered flit is sent to the Odd bank of a downstream router, and similarly, an even numbered flit to the Even bank through a Multibank Buffer Arbiter (MBA) that has one input port and two splitted output ports.

The goal of this multibank buffer scheme is to enable the incoming consecutive flits to be written to different banks simultaneously to effectively hide the multicycle write latencies of STT-MRAM. To achieve this goal, two MUXes and one simple latch are used for the MBA as shown in Figure 8(b). Each MUX has two inputs: one input is connected to the communication link from the upstream router, and another to the simple latch inside the router. The simple latch is located at the front of the input buffer and functions as a temporary buffer. It holds an incoming flit for a cycle and dispatches the latched flit to its original target bank at the very next cycle. \( I_{\text{clk}} \) and \( M_{\text{clk}} \) are control signals originating from the control block in the input buffer, which represent the hold/release signals for the latch. The area overhead for this logic is negligible as compared to the buffer, and already added to the logic area controlling refresh/read/write pointers. An incoming Odd flit, for instance, is directly written to the Odd bank during the first cycle, and then during the next cycle, the latched flit is sent to the same Odd bank, thus completing its 2-cycle write process\(^4\). Similarly, a subsequent incoming Even flit follows the same process, but uses the other bank. Through this, without the need of any additional SRAM buffer as in the Hybrid buffer \([27]\), we can seamlessly pipeline incoming consecutive flits to the input buffers of a downstream router.

Note that, in case of very light loads, an incoming flit might experience write delays in the STT-MRAM buffer, increasing zero-load latency, which results in degraded NoC performance.

---

\(^4\)Since it takes multicycles to write a single flit to a target buffer, there could be a potential glitch due to a momentary transient pulse (noise), or clock skew between communicating elements. These issues can be addressed by sizing the MUX, overlapping clock or duplicating input signal \([21]\).

Figure 8: Multibank STT-MRAM Buffer

To avoid this, we incorporate the buffer bypassing logic \([51]\), widely used in NoCs for power-performance efficiency \([1, 37]\). Accordingly, when a flit arrives at an empty buffer, the flit heads straight to switch arbitration and gets sent directly to the crossbar switch, thus circumventing STT-MRAM input buffers. The latch inside a router serves as a bypass latch for the consecutive pipelining between the flit arrival and crossbar traversal. Therefore, the zero-load latency of the STT-MRAM router becomes comparable to that of the SRAM based router.

In general, to hide \( n \)-cycle write latency, the STT-MRAM buffer scheme requires \( n \) MUXes for \( n \) split banks with \( n - 1 \) latches inside a router as shown in Figure 12. The increase of \( n \) can negatively affect the performance and area overheads of the STT-MRAM buffer. Note that \( n \) is the ratio of the STT-MRAM write latency to the clock cycle time of the NoC clock. As technology advances, we expect the write latency to be reduced as described in Section 2.2, while at the same time the NoC clock frequency increases. Therefore we do not expect \( n \) to increase drastically in the near future, hence keeping the proposed scheme feasible. In our analysis, when \( n \) stays within 5, we observe negligible performance degradation (less than 1%) with increased extra logic area. The detailed analysis of the impact of large \( n \) is discussed in Section 5.4.

Figure 9 shows an example data flow for flits from an upstream router during 3 consecutive clock cycles. Initially, the control of both MUXes, denoted as MUX0 and MUX1, is assumed to 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.

- **Cycle 0**: The input signal of both MUXes is set to 1 (IN1). This is the first write cycle for an incoming flit, Flit1. Flit1 is sent to the Odd bank input buffer of the downstream router through IN1 of MUX0, and at the same time, Flit1 is stored in the simple latch (\( \text{T} \)).
- **Cycle 1**: The input signal of both MUXes has changed to 0 (IN0). The output of MUX0 is Flit1 which was previously
latched, and Flit 1 is dispatched from the latch to its original target bank (Odd bank), and thus completing its second write cycle(2)). Also, this is the first write cycle for a subsequent incoming flit, Flit 2, to the Even bank input buffer. While Flit 2 is transferred to the Even bank through IN0 of MUX1, it is simultaneously stored in the simple latch(3).

- Cycle 2: The input signal of both MUXes is switched back to 1 (IN1). Like the previous logic, this is the second write cycle of Flit 2 from the latch to the Even bank (4), and the first write cycle for the incoming flit, Flit 3(5).

Note that, at low loads, flits arrive at the input buffer intermittently. In this case, the arriving flit bypasses the input buffer, or unless the buffer is empty, the STT-MRAM buffer directs the flit to either Odd or Even bank based on a one-bit flag indicating the next bank. This ensures that incoming flits are placed in a VC without leaving unused buffer slots in banks. This also ensures sequential reads by maintaining the FIFO properties of input buffers.

4.2.2. Read/Write and Refresh Logic

Unlike the conventional SRAM input buffer that requires a read and a write pointer for read and write operations per VC (Figure 10(a)), the proposed multitype STT-MRAM buffer, assuming 2-cycle write latency, requires dual write pointers, Wr_ptr (Odd) and Wr_ptr (Even), and a single read pointer, Rd_ptr, per VC as shown in Figure 10(b). The corresponding VC control logic generates proper read and write pointer values for handling flits in a timely manner. To be specific, initially, as shown in Figure 11(a), one of the write pointers, Wr_ptr (Odd), points to the tail of the Odd bank, and Wr_ptr (Even) points to the tail of the Even bank, and the read pointer, Rd_ptr, points to the head of the buffer. For a write operation (Figure 11(b)), the incoming flit is written to the location pointed by the tail pointer in one of the banks. For a read operation (Figure 11(c)), the flit pointed by Rd_ptr is read out and dispatched to the crossbar. Note that STT-MRAM read latency is comparable to that of SRAM and thus no delay occurs for the read operation. The refresh pointer, Refresh_ptr, shown in Figure 11(d), moves according to the refresh logic which is described in Section 4.3.

4.3. Nonvolatility-Relaxed STT-MRAM Buffer

In this section, we propose cost-efficient dynamic buffer refresh schemes to avoid flit dropping, which occurs when flits overstay a given retention time.

4.3.1. Dynamic Buffer Refresh Schemes

To avoid flit losses incurred by the relaxed nonvolatility of STT-MRAM buffers, we employ two different refresh schemes through which refresh operations are adaptively triggered for expiring flits which almost reach the end of a given retention time. Basically, these refresh schemes require a n flit-deep refresh buffer per each physical channel (PC) for n-cycle write latency. During a refresh operation, the target flit is read out from the input buffer into the refresh buffer, and then written back to its original FIFO buffer. If a read request comes before refresh finishes, the flit is directly returned from the refresh buffer. The refresh buffer is also used as a read buffer to compensate for n-cycle write latency of STT-MRAM, and the refreshes are seamlessly pipelined to allow consecutive refreshes. Note that the refresh buffer can be made using either an external refresh buffer (ERB) of SRAM, or an internal refresh buffer (IRB) of STT-MRAM. The ERB requires two writes: an initial write to the ERB and another subsequent write to the original FIFO buffer. To decrease such redundant write overheads in the ERB, we can relocate the refresh buffer inside the FIFO buffer and when a flit needs to get refreshed, the flit is moved to the IRB, and the IRBs keep progressing through the FIFO buffer as flits get refreshed. Note that the IRB is hidden from the upstream router to prevent the VA stage in the upstream router from allocating these buffers to flits. It is also noted that the impact of refresh overheads on the network throughput is negligible due to such seamless refresh operations. Also, since flits arrive at VCs at different points of time, no two flits have the same timestamp across all VCs in the same PC. Accordingly, we only need a single refresh buffer per PC for the refresh operation, and we assume a single flit can be refreshed per cycle in each PC.

1) Simple Refresh Scheme is a straightforward refresh scheduling algorithm, which simply forces a refresh operation
to be performed sequentially in a VC when a flit queued in the head of the VC reaches a certain refresh threshold, $T_0$. $T_0$ is predefined statically, expressed as clock cycles, such as 100 cycles in 2 GHz clock frequency, and checked periodically by a refresh logic. Based on the FIFO properties of the buffers, the foremost flit is always the oldest one in a VC. This scheme guarantees that every subsequent flit stored in the same VC gets refreshed in a proactive way within the retention time interval of STT-MRAM, thus no flits get dropped in any case. The simple refresh scheme, however, unnecessarily wastes significant dynamic power at high loads where lots of flits circulate among routers. This is because it considers only the age of the foremost flit inside a VC, thus refreshing a group of newly arrived flits too early. To minimize such an inefficiency in the simple refresh, we propose a more fine-grained refresh policy below.

2) Global Counter (GC) Refresh Scheme selectively triggers the refresh operation based on the estimated age of an individual flit per VC. To monitor the age of each flit, we add a refresh pointer, Refresh-ptr, shown in Figure 11, which is controlled by a VC refresh logic shown in Figure 10(b). Initially, the refresh scheduler makes the refresh pointer point to the flit queued in the head of a VC and moves it toward the tail of a VC one flit at a time whenever the pointed flit gets refreshed as shown in Figure 11(d). To decide the refresh timepoint, we adopt a per-router GC, which serves as a reference point for the refresh logic to decide if it needs to trigger refresh operations to keep the validity of flits.

In this scheme, STT-MRAM cell’s retention time is divided into $N$ periods, and each period is $T$ cycles long. A per-router GC is used to maintain the countdown to $T$ cycles. At the end of every $T$ cycles, the value of GC is increased by one, and loops over after the given retention time. Figure 13 shows an example of 2-bit GC refresh scheme. When a GC reaches a certain level, a buffer may have reached the maximum retention time, and hence needs to be refreshed. In Figure 13, the GC is updated at the end of every $T$ cycles ($T = 50$), and when a flit arrives at a buffer, the value of the current GC (00, 01, 10, 11) is copied to the flit’s Arrival Counter (AC). At the end of $T$ cycles, the AC value of each flit is compared with the (GC+1) value to see if the current flit needs to be refreshed. When the GC is 01, for example, all flits having AC equal to 10 get refreshed one by one per cycle. This is equivalent to refreshing flits that stay at the buffer for around 150 cycles. Note that the per-flit AC value is assigned only when a flit arrives at a buffer, and unchanged until the flit gets dispatched. Note that as the interval of a period gets decreased (more bits are assigned to a GC), less refresh operations are performed since a refresh is triggered based on a more fine-grained clock counter, thus saving more dynamic power, which is detailed in Section 5.3. A larger bit counter allows more time for a flit to stay in the buffer before applying any refresh. This gives the buffer more opportunity to hold flits inside at the cost of maintaining a counter with more bits. Our experimental results show that a GC suffices to detect the expiration time of the flits without significantly affecting performance, which is described in Section 5.2.

Figure 14 compares the refresh overheads of three different refresh schemes (simple, GC (2bit), and GC (3bit)) combined with different SAs (Round-Robin (RR) and Age-based). With Age-based SA, oldest flits are always chosen over others, and then dispatched to the crossbar before getting expired, thus it is inherently much more effective in reducing the number of refreshes compared with RR where flits are chosen in a Round-Robin fashion regardless of their relative ages. Age-based SA triggers less refreshes compared with RR by 73.6%, on average. As a result, among the six different combinations, GC(3bit)-Age is the most effective in the reduction of refreshes, and especially, when compared with the worst performing SIMPLE-RR, GC(3bit)-Age triggers less refreshes by 93.3%.

In this way, we can save significant dynamic power compared with the aforementioned simple refresh and periodic DRAM-style refresh [44] where refresh operations are performed for all STT-MRAM blocks in sequence regardless of its data contents, introducing many unnecessary refreshes. The total refresh power consumed by the GC refresh scheme is reduced by up to 39.6% and 96% compared with the simple refresh and DRAM-style refresh, respectively, which is detailed in Section 5.3.
5. Performance Evaluation

5.1. System Configuration

A cycle-accurate NoC simulator is used to conduct the detailed evaluation of the proposed schemes. It implements the pipelined router architecture with VCs, a VC allocator (VA), a switch arbiter (SA) and a crossbar. The network is equipped with 2-stage speculative routers with lookahead routing [17]. The router has a set of VCs per input port. Each VC contains a k-flit buffer with 16B flit size. In our evaluation, we assume that v is 4, while k may vary with different buffer configurations. A dimension order routing, XY, is used with wormhole switching flow control in an (8x8) 2D-mesh. A variety of synthetic workloads are used to measure the effectiveness of the STT-MRAM buffer schemes: uniform random (UR), bit complement (BC) and nearest neighbor (NN). To evaluate the proposed schemes under realistic environments, we also run PARSEC [3] parallel benchmarks via the trace readers in Netrace [23] that is incorporated into our NoC simulator. Table 1 specifies the detailed CMP configuration.

To estimate the power, area, and timing of SRAM/STT-MRAM routers operating with 1 V supply voltage in 2 GHz clock frequency, we modified an open source NoC router RTL model [2] and synthesized in Synopsys Design Compiler with a TSMC 45nm technology library. SRAM/STT-MRAM parameter values in Table 2 are obtained through the STT-MRAM analyses described in Section 2.2 and based on relevant STT-MRAM literatures [20, 27]. Note that the unit of the leakage power is mW per 1-flit buffer. Throughout this paper, the sizes of the SRAM and STT-MRAM buffers, defined by the number of flits per VC, are denoted by SRAM# and STT#, respectively. As stated in Section 2, STT-MRAM basically provides 4 times more buffer capacity compared with SRAM under the same area budget (SRAM4 is equal to STT16). For the STT-MRAM buffer schemes, however, due to the extra area overhead incurred by additional circuitry for the MBA shown in Figure 8, 2.58% of buffer spaces get sacrificed under 2-cycle write latency. Thus, STT-MRAM can provide approximately 3.5 times more buffer capacities than the conventional SRAM buffer (SRAM4 is equal to STT14). The detailed area analysis is given in Section 5.4.

5.2. Performance Analysis

Figure 15 shows performance results of three different buffer configurations: the SRAM buffer, the Hybrid buffer, and the proposed STT-MRAM buffer under UR, BC, and NN traffic patterns. All results are measured under the same area budget, SRAM4 per VC, for input buffers. The Hybrid buffer can accommodate 7 flits per VC, consisting of SRAM3 and STT4, which is an optimal hybrid design suggested in [27], while the STT-MRAM buffer accommodates 14 flits per VC. In all cases, on average, the STT-MRAM buffer shows better throughput by 19.9% for UR, 23.9% for BC, and 21.2% for NN compared with the SRAM buffer, and 5.1% compared with the Hybrid buffer across different traffic patterns. These results indicate that the potential performance degradation caused by the multicycle write latencies of STT-MRAM can be offset by the increased buffer size and the proposed multibank STT-MRAM buffer scheme, thus resulting in significant performance improvement.

We also evaluate the STT-MRAM buffer under various topologies: Concentrated Mesh (CMesh), 2D-Torus, and Flattened Butterfly [30]. Figure 16 shows that the STT-MRAM buffer helps increase throughput in CMesh, 2D-Torus, and Flattened Butterfly by 25.9%, 20.2%, and 10% compared with the SRAM buffer, and 5.7%, 9.0%, and 5.2% compared with the Hybrid buffer, respectively.

Figure 17 shows speedups relative to the SRAM baseline with PARSEC benchmarks. We assume SRAM4 per VC as an area budget, the same as a cache block size. The average network load in PARSEC benchmarks is low, but because they exhibit bursty communication and have congestion periods (the time period when the average ratio of buffer occupancy in injection ports is above a threshold, which is set to 75% in our study), our scheme contributes to improve NoC performance.
In Figure 17, the relative performance improvement of the proposed scheme over the SRAM baseline is not comparable to those shown in Figure 15 (vips gets 11.0% improvement while blackscholes gets 9.7% and dedup gets 7.0%), and the STT-MRAM and Hybrid buffers show similar performance. However, when we analyze the performance in the congestion period, the STT-MRAM buffer outperforms the SRAM and Hybrid buffers by 15.2% and 9.3%, on average, respectively.

We perform sensitivity analysis by varying packet lengths, area budgets, and number of VCs as shown in Figure 18 to examine their effects on NoC throughput. Figure 18(a) shows the normalized throughput improvement with different packet lengths: 4, 8, 12, and 16 flits per packet. All results are normalized to that of baseline SRAM buffer with 4 buffer slots per VC. It clearly shows that the STT-MRAM buffer works better as packet length increases. The longest packet consisting of 16 flits (PKT_16), shows the biggest performance improvement up to 30% in the STT-MRAM buffer over baseline SRAM. This is because when the buffer capacity is not big enough to accommodate a whole packet, the packets in transit tend to spread across multiple nodes, thus impeding subsequent packets from proceeding to their destination, which results in significant performance degradation. NoC throughput is also mutually affected by two important factors: buffer depth and number of VCs per PC. Figure 18(b) shows the impact of buffer depth on throughput improvement with five different area budgets: SRAM3, SRAM4, SRAM5, SRAM6, and SRAM7. The more we increase default area budget, the deeper buffer depth we can provide, thus improving network throughput. Across the given budgets, the STT-MRAM buffer shows the highest throughput improvement. Under the smallest budget (SRAM3), the STT-MRAM buffer enhances throughput by 23% and 11% over the SRAM and Hybrid buffers, respectively. However, deepening the buffer depth does not always yield tangible throughput improvement as shown in the largest budget (SRAM7). This is mainly because Head-of-Line (HoL) blocking occurs when many packets contend for router resources (limited number of VCs), but the increased buffer depth does not alleviate this problem. As shown in Figure 18(c), increasing the number of VCs per PC is an effective way of improving network throughput further because it allows more packets to share the same PC, thus reducing HoL blocking.

5.3. Power Analysis

Power is one of the critical issues in designing NoC. We measure the power consumption of the proposed multibank STT-MRAM buffer scheme against the SRAM and Hybrid buffers, and evaluate the effectiveness of the different buffer refresh schemes described in Section 4.3.1.

Figure 19(a) compares the dynamic power consumption of the SRAM, Hybrid, and STT-MRAM buffers with different packet injection rates under UR traffic. All results are normalized to that of the SRAM buffer. The first and second bars indicate the SRAM and Hybrid buffers and, in particular, the STT-MRAM buffer is evaluated based on different refresh schemes (Simple or 3-bit Global Counter (GC)), and SAs (Round-Robin (RR) or Age-based) to quantitatively measure their effectiveness in reducing refresh power overheads, and find out the most power-efficient combination\(^5\). Note that the refresh power overheads affect the dynamic power consumption in NoCs, and are

\(^5\)See Figure 14 for details on the efficiency between refresh schemes.
increasing proportionally to the number of refreshes performed. Thus, to achieve a power-efficient NoC, it is necessary to employ a buffer refresh scheme that triggers less refreshes. In Figure 19(a), the baseline SRAM consumes the least normalized dynamic power because the Hybrid and STT-MRAM buffers require higher write energy compared to that of the SRAM (see Table 2). The Hybrid buffer consumes 1.7 times and 1.4 times more dynamic power, on average, compared with the SRAM and STT-MRAM buffers each. This is mostly due to the frequent migrations from SRAM to STT-MRAM inside the Hybrid buffer, and a higher write energy associated with a high retention STT-MRAM, 10 ms, in the Hybrid buffer, compared to that of the multibank STT-MRAM buffer. For the STT-MRAM buffer, GC(3bit)-Age consumes the least dynamic power across different combinations of refresh schemes. Specifically, compared with the SIMPLE-RR and SIMPLE-Age, GC(3bit)-Age consumes less dynamic power by 15.4% and 8.0%, respectively at high injection rates (> 0.35).

Figure 19(b) compares the total power consumption of routers with different buffer schemes. The total router power includes dynamic and leakage power of all routers across the network. On average, there is 17% improvement in total router power going from baseline SRAM to STT-MRAM buffer design. With our proposed buffer refreshment schemes, although there is an increase in the dynamic power, we consistently observe efficiency in total router power consumption of the proposed STT-MRAM buffer. This is attributed to the fact that the fraction of dynamic power to the total power is not significant compared to the very high leakage power [29, 54]. In this context, due to the power hungry nature of SRAM, the SRAM and Hybrid buffers consume significantly more power than the STT-MRAM buffer. Among the three different combinations of the STT-MRAM refresh schemes, GC(3bit)-Age consumes less total router power by 8.4% and 14.7%, compared with the SIMPLE-RR and baseline SRAM, respectively. Also, the Hybrid scheme consumes 9.8% and 32.5% more total router power, on average, especially at high injection rates (> 0.3), compared to the SRAM and STT-MRAM schemes. This is because of the migration power overheads, high STT-MRAM write energy, and high SRAM leakage power in the Hybrid buffer.

Figure 20 compares the total router power consumption with PARSEC benchmarks. Among the three different schemes, the STT-MRAM router is the most power-efficient, consuming 17.8% and 44.9% less power compared with the SRAM and Hybrid routers, respectively. Blackscholes consumes the least total power in the STT-MRAM router by 18.7% and 46.4% compared with the SRAM and Hybrid routers.

5.4. Impact of Write Delays of STT-MRAM

For our scheme, STT-MRAM write latency is an important factor affecting the overall NoC area and performance. Till now, in our experiments, we assume STT-MRAM has 2-cycle write delay with a density of 3.5 times SRAM, but as we increase the write latency further, the extra logics, such as MUXes and latches, hiding the multicycle writes need to be added. Due to such additional spaces taken up by the extra logics in the STT-MRAM buffer, STT-MRAM is given relatively less area in the given buffer space. Specifically, when n (write delay) equals 2, initial single router area, its buffer area (per input port), and extra logic area (per buffer) are 106,709 µm², 14,762 µm² (A), and 689 µm² (B), respectively, where the effective buffer area is 14,073 µm² (A-B). As we increase n, while keeping per router area budget intact, extra logic area increases by approximately 7.5% per additional write latency, thus leading to decreased...
effective buffer space per input port (Figure 21). Across the different write latencies of STT-MRAM (3, 4, and 5), we observe negligible performance degradation (less than 1%) under UR traffic. This is because STT-MRAM still provides enough buffer space to sustain the network bandwidth. However, the performance becomes equal to or less than that of SRAM baseline when the STT-MRAM buffer has similar capacities with the SRAM buffer due to the reduced density. In our configuration, this occurs when the STT-MRAM write latency is 9 or more cycles.

5.5. Comparison with Other NoC Techniques

There have been a few studies to improve performance with limited buffer resources in NoC design [39, 36]. We compared the power and performance benefits of our scheme with them. BLESS [39] reduces buffer power and area overheads by eliminating router buffers, and handles network contention by deflecting contending flits to any free output port. In our evaluation, the performance overheads of BLESS outweigh its gains due to the increased allocator complexities that avoid livelocks, and the extra packet overheads, where flits in a packet contains routing information to be independently routed to the destination. BLESS saves significant router area by eliminating buffer spaces, but the frequent deflections of BLESS at high loads consume significant dynamic power, and leads to early network saturation as shown in Figure 22. On the other hand, the SST-MRAM router provides higher throughput by 54.1% than that of BLESS, and is more power-efficient at flit injection rates greater than 22% compared to BLESS. WPF [36] proposes a bandwidth-efficient, fully adaptive routing scheme in VC-limited NoCs where short packets dominate. WPF makes packets traverse all minimal paths, thus enhancing routing flexibility, and provides deadlock avoidance techniques which allow non-empty VCs to be re-allocated, achieving high VC utilization. Under the same area budget, as shown in Figure 22, under UR traffic, the SST-MRAM router shows better performance by 8.5% than that of WPF due to the high density buffer properties of SST-MRAM.

6. Related Work

Many prior studies have adopted STT-MRAM to provide power/performance-optimized architectural supports and density-optimized secondary storages. Guo et al. [20] detailed STT-MRAM based architectural techniques to offer power-efficient and scalable microprocessors. Goswami et al. [19] proposed STT-MRAM based GPGPU architectures and hybrid shared memory for power-performance optimizations. In [29, 44, 48], the data retention time of STT-MRAM has been carefully adjusted to achieve better write performance and reduced write energy for caches in CMPs. However, the cache based schemes cannot be directly applicable to NoCs since they are designed for memories having longer data residence time and larger capacity compared to FIFO buffers. Power-gating [6] and bufferless NoC [16, 22] have been introduced to reduce router power consumption. Power-gating is a circuit-level technique mitigating the static power consumption of NoCs by cutting off power temporarily. However, due to frequent state transitions and unavoidable wakeup time delays, as described in [6], power-gating rather consumes more power at high load, and has higher average packet latency at both low and high load compared to a non-power-gated NoC, and such overheads increase as network scale grows. Bufferless NoC eliminates buffers, thus the peak network throughput is reduced, and as stated in [16], it has higher packet latency overall, resulting in degraded performance. Also, although network power is often significantly reduced at low-to-medium load, bufferless NoC consumes more power as the network becomes congested compared to a buffered NoC mostly due to increased link power from frequent deflections. However, in this work we provide high bandwidth NoC with low power consumption even at high network load using the proposed STT-MRAM NoC router.

7. Other Applications and Future Work

The multibank STT-MRAM buffer scheme can be used in multiple domains such as instruction queue, reorder buffer, and prefetch buffer. Since instructions are generally used up quickly, large instruction queues allow for better instruction level parallelism [40]. In this context, we will explore using our scheme as a worthy prospective in terms of power consumption and queue length. There are several studies dealing with the efficiency of reorder buffers [33, 32], and exploring the lifetimes of variables in programs [35], which show that a large portion of the variables whose values are held in the reorder buffers are short lived. This leads to a mixture of variables with irregular lifetimes. We plan to examine a hybrid SRAM/STT-MRAM buffer scheme which accounts for the variation in retention times required. We will explore utilizing STT-MRAM buffers as an attractive alternative to prefetch buffers [7] for saving power and providing larger buffer space since prefetched data do not need to be cached for a long time. We also plan on tackling the challenges of having data retention times tuned to the timeliness of prefetching so as to make our design feasible.

8. Conclusions

In this paper, we have proposed a novel NoC router input buffer design with STT-MRAM, and achieved significant throughput improvement with efficient power consumption. The high density of STT-MRAM facilitates accommodating larger buffers compared with SRAM under the same area budgets.
Through the multibank STT-MRAM buffer design combined with power-efficient refresh schemes, the long write latency of STT-MRAM is effectively hidden while consuming less power. Simulation results show performance improvement of 21.6%, and 17% savings in the total router power compared with a conventional SRAM based NoC router.

References


