THROUGHPUT-EFFICIENT NETWORK-ON-CHIP ROUTER DESIGN WITH STT-MRAM

A Thesis

by

SAGAR NARAYANA

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

December 2012

Major Subject: Computer Engineering

ABSTRACT

As the number of processor cores on a chip increases with the advance of CMOS technology, there has been a growing need of more efficient Network-on-Chip (NoC) design since communication delay has become a major bottleneck in large-scale multicore systems. In designing efficient input buffers of NoC routers for better performance and power efficiency, Spin-Torque Transfer Magnetic RAM (STT-MRAM) is regarded as a promising solution due to its nature of high density and near-zero leakage power. Previous work that adopts STT-MRAM in designing NoC router input buffer shows a limitation in minimizing the overhead of power consumption, even though it succeeds to some degree in achieving high network throughput by the use of SRAM to hide the long write latency of STT-MRAM.

In this thesis, we propose a novel input buffer design that depends solely on STT-MRAM without the need of SRAM to maximize the benefits of low leakage power and area efficiency inherent in STT-MRAM. In addition, we introduce power-efficient buffer refreshing schemes synergized with age-based switch arbitration that gives higher priority to older flits to remove unnecessary refreshing operations. On an average, we observed throughput improvements of 16% on synthetic workloads and benchmarks.

To my loved ones

ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. Eun Jung Kim for her guidance and support throughout the course of this research. I would also like to thank Dr. Paul Gratz and Dr. Peng Li for their valuable feedback.

Thanks also go to my friends and team members at High Performance Computing Laboratory, especially Hyunjun Jang, Baik Song An, Rohan Kansal.

I extend my gratitude to my room-mates and friends for making my time at Texas A&M University a great experience.

Finally, thanks to my family for their encouragement and support.

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 higher degree of parallelism through chip multiprocessors (CMPs) by fabricating multiple cores and cache banks in a single chip. Technology trend is also moving to many-core environments where more than hundreds of processor cores are integrated in a single die. Along with this trend, the size of interconnection networks also increases to provide proper communication paths for processors and on-chip caches. It is projected that network overheads will be more dominant than computation power in overall system performance. Shared buses are the simplest solutions to interconnection networks because they can be easily implemented to provide connectivity to all nodes by connecting them in a single medium. They cannot be an efficient solution, however, in large-scale systems in that they become easily congested due to the limited scalability. Instead, switch-based Network-on-Chip (NoC) is considered as an efficient infrastructure for many-core CMP systems. NoC interconnects communication nodes efficiently in a flexible and scalable manner to achieve better performance. Resources available for NoC, however, are tightly limited compared to off-chip interconnects due to the inherent constraints of relatively small area and power budget of CMPs. Even though the area budget in a chip increases as the feature size keeps shrinking, the budget available to interconnects should be carefully allocated because a huge portion should be assigned to processor cores and on-chip caches. Therefore, new and innovative NoC design is required to provide better performance and high efficiency under a highly constrained on-chip interconnects.

Among all components in NoC, input buffers takes up significant portion of area and power budgets. Input buffers are commonly implemented with SRAM because it guarantees fast access speed for read and write operations. However, non-negligible area cost and leakage power consumption of SRAM is giving lots of pressure on scalable NoC design. In that regards, Spin-Torque Transfer Magnetic RAM (STT-MRAM) is considered as a promising next generation memory technology that can replace the existing SRAM in that it offers numerous beneficial features to memory design, such as near-zero consumption of leakage power and high integration density compared to the existing SRAM [6]. In addition, STT-MRAM provides high endurance that tolerates frequent write accesses to the memory. The frequency of memory operations, especially write accesses, is much higher in NoC input buffers than in caches or off-chip memories, which make STT-MRAM more attractive, compared to other memory technologies such as Phase Change Memory (PCM) or Flash. However, the weaknesses of STT-MRAM, the long latency and high power consumption in write operations, should be properly handled so that STT-MRAM can be utilized to its full potential in NoC.

1.1 Motivation

Most recent work [7] that attempts to adopt STT-MRAM in designing NoC input buffers suggests a new hybrid design of input buffers that combines both SRAM and STT-MRAM to effectively hide the long write latency of STT-MRAM. Even though it shows a noticeable improvement in overall network throughput in different workloads and topologies, however, it is not successful in making the best use of the advantages of STT-MRAM in terms of power consumption because of the high leakage power inherent in SRAM which greatly weakens the benefits of STT-MRAM. Moreover, the frequent migration of flits from SRAM to STT-MRAM consumes significant amount of dynamic power.

In this thesis, we propose a novel input buffer design which is highly efficient in terms of both network performance and power consumption. It relies solely on STT-MRAM without requiring SRAM to boost the advantages of STT-MRAM, near-zero leakage power and high density. We believe this is the first work to design NoC input buffers purely based on STT-MRAM. Motivated by the observation that relaxing the non-volatility of STT-MRAM helps to achieve better write performance, we redesign an STT-RAM memory cell so that each write operation can complete in less than 1ns, corresponding to 2 cycles in 2GHz clock frequency, through sacrificing data retention time.

We still have a huge margin of retention time since in NoC input buffer data stay extremely short period of time than other memory or storage devices. Considering that the 2 cycles of write latency still worsen the performance compared to existing SRAM, we also propose a novel input buffer design that enables a simple flop to serve as temporary data storage. Through this, the long write latency of STT-MRAM can be effectively hidden without incurring the degradation of performance while consuming much less power.

Even though the reduced data retention time of STT-MRAM is still enough to hold data safely in most cases, we still need to consider the situation when data stay in STT-MRAM input buffers longer than the given retention time, which leads the data to become corrupted. To prevent such data corruption, we propose power-efficient buffer refreshing schemes combined with age-based switch arbitration which gives priority to old flits staying in the buffer. Thus, the longer a flit stays in the buffer, the higher it has a chance to be chosen in arbitration stage, so the possibility of data corruption can be significantly reduced. In practice, it is observed that the age-based switch arbitration (SA) considerably reduces the number of flits dropped across the network compared to conventional round robin SA policy.

1.2 Outline of Thesis

In Section 2, we include the basics of On-chip networks and non-volatile memories.

In Section 3, we look at several works where NVMs are used in processors, caches and memories. We also look at the previous work on buffer management in NoC.

In Section 4, we model the STT-MRAM and design STT-MRAM memory cell with decreased retention time and 2 cycles of write latency. We discuss the problems of packet dropping introduced by decreasing retention time.

In Section 5, we describe in detail purely STT-MRAM based On-Chip Router Architecture and techniques to prevent packet dropping.

In Section 6, we include experimental setup and performance evaluation.

Lastly, Section 7 contains conclusions of the work.

2. BACKGROUND

2.1 Network-On-Chip

As the technology continues to scale, more and more system components are packed on a single chip. Thousands of processor cores can now potentially be put on a single chip, and providing communication between them is non-trivial. The trend has now moved towards manufacturing System-On-Chip (SoC) which contains a previously large, distributed, on-board system on a single chip. This trend has posed new challenges in chip design.

To provide a perspective for one of the major challenges in SoC design, and motivation for introducing NoC, we can look at the components of SoC as containing two major subdivisions: Components which provide computation and those which provide communication between the components. When technology scales, communication wire delays don't scale the same way as gate delays. This asymmetry makes communication more costly and to have higher impact on performance at lower process technology. It is also impractical and costly to provide a global synchronous clock to all the components in such a design. GALS: Globally Asynchronous Locally Synchronous System design is one of the approaches to solve this problem.

In a very small system, communication between several processing cores can be provided either by point to point links or shared buses. However, both the approaches don't scale to a large number of cores. When the number of processing cores increases, the number of point to point links increases exponentially. Shared buses pose a different problem. Increasing the number of components connected to a single shared bus deteriorates bus speed and requires slower clock owing to the capacitance introduced by all the components. The bus arbitration design also takes a hit, since it needs to be redesigned for larger number of components. All these issues make shared bus impractical for very large number of cores.

NoC is a novel solution which is a scalable communication network inside a chip. At a high level, NoC borrows concepts from off-chip networks. However, the tradeoffs are different since off-chip and on-chip resources have different requirements. NoC design scales linearly with the number of cores making it design friendly among others. The various aspects of NoC are outlined below.

2.1.1 Topology

A typical NoC consists of several processor cores/cache banks/components attached to a NoC router via a network interface (NI). NoC routers communicate with each other via on-chip network links. Topology of NoC defines how the routers and links are placed and connected to each other. Topology of NoC directly affects throughput and latency of the network. It also dictates the scalability of the NoC design.

A few common topologies are shown in Figure 1. Mesh and Torus topologies are very popular for being simple and scalable. While mesh is easier to layout on a chip, it is not symmetric. In Torus, the wrap-around links will be longer on a 2-d chip layout. This can be avoided by a folded torus. These topologies can be called uniform in the sense that all the routers have similar communication resources available to them.

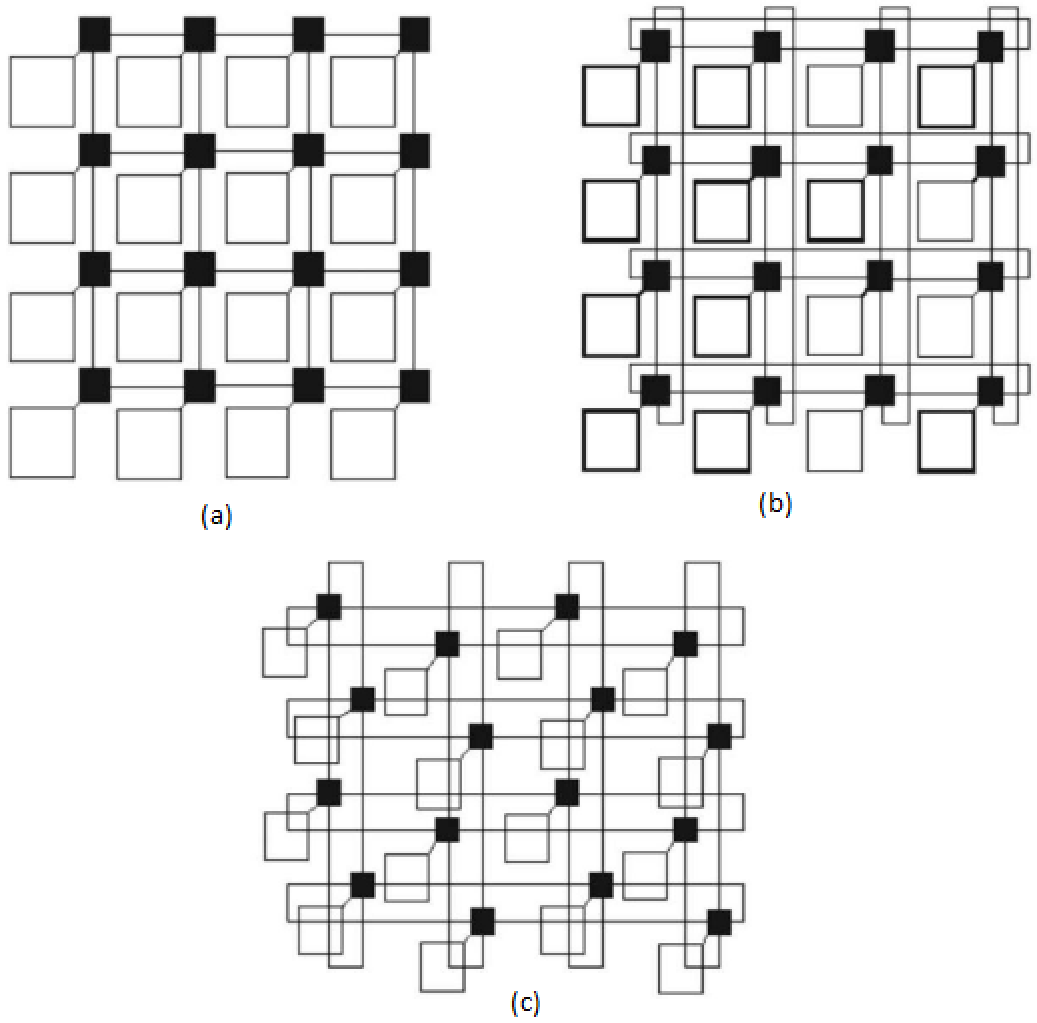


Figure 1. Common NoC topologies. (a) Mesh (b) Torus (c) Folded torus

2.1.2 Routing

The routing algorithm decides the path a packet should take to travel from source to destination. Depending on the topology like mesh, there can be multiple paths between a source and destination. Thus, routing algorithm has an effect on overall throughput and latency. An ideal routing algorithm will distribute the load across all the channels for any kind of traffic. It also has an effect on power and area since a complicated routing algorithm is costlier to implement.

Routing algorithms can be divided into deterministic, oblivious and adaptive routing algorithms.

As the name suggests, deterministic routing algorithms always take the same path for a given source and destination. This makes these algorithms far from ideal since always the same path is chosen irrespective of the traffic and they don’t do any kind of load balancing. However, they are very simple and inexpensive to implement. An example is dimension order routing for mesh networks. For some topologies, deterministic routing performs as well as any other routing algorithms.

Oblivious routing is also simple to implement. The routes for packets are chosen irrespective of the state of the network. In other words, they are oblivious to the state of the network. Examples are Valiant’s randomized routing algorithm and minimal oblivious routing algorithm. In Valiant's randomized routing algorithm, to route a packet from source and destination, any random intermediate node is chosen and is routed from there. This destroys any locality of the traffic pattern. However, it achieves load balancing. In minimal oblivious routing, only routes among a set of minimal routes are chosen. This maintains the locality and achieves good load balancing.

Adaptive routing algorithms use the state of the network to make routing decisions. This should make them very flexible and powerful, however, they are also costly to implement. It is costly to get global state of the network at every router, so, most practical adaptive routing algorithms use local state of the network by the way of queue length, credit backpressure, etc. Because of this partial blindness, not all adaptive routing algorithms perform better than oblivious routing algorithms. Making these algorithms deadlock free is also a major concern.

2.1.3 Router Architecture

A typical NoC router consists of several data path and control components to implement the functionality. Figure 2. shows a virtual channel router [32]. It contains input units and output units separated by a switch. There is an input unit for every input port and similarly for output units. The switch connects input units to output units. A packet is composed of flits, and each input unit has a set of flit buffers to hold flits of a packet. When a flit arrives at the input port, it is written to the appropriate input unit buffers. Route Computation (RC) decides which output port the packet should be forwarded to. Once the output port is decided, Virtual Allocation (VA) decides which output virtual channel the packet should be assigned to. Switch Allocator then allocates a time slot for the slit to traverse the switch during Switch Traversal.

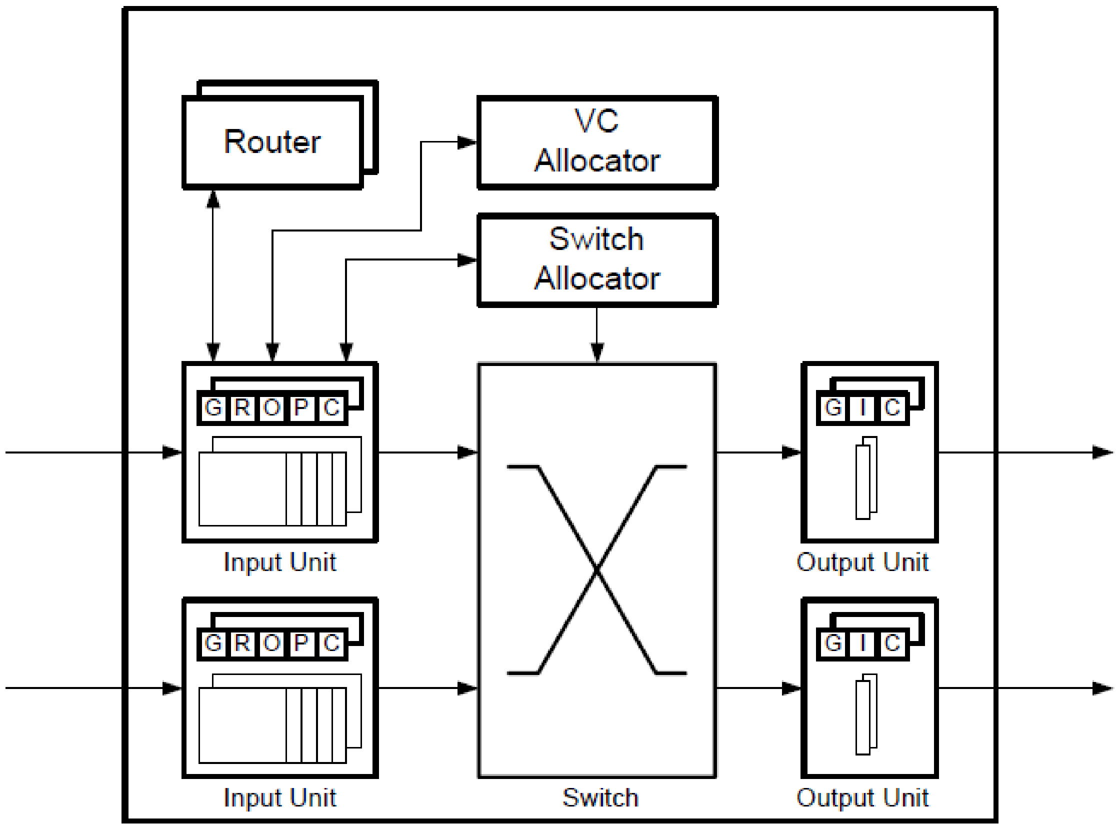


Figure 2. Block diagram of virtual channel router

All the routing operations can be pipelined and overlapped. Pipelining of a typical VC router is given in Figure 3. (a) [32]. A detailed working of the pipelining can be found in [32]. Note that, there can be stalls in the pipeline: packet stalls and flit stalls. With speculation and look ahead, it is possible to reduce the pipeline stages by performing many routing operations in parallel. Figure 3. (b) shows one such speculative router pipeline [32].

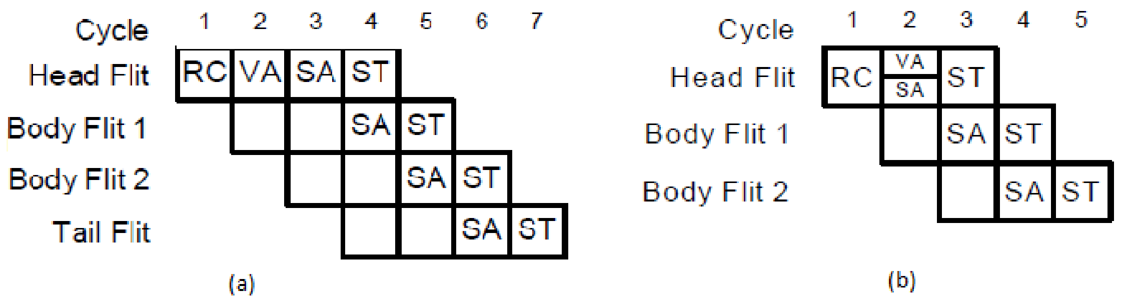


Figure 3. NoC router pipeline (a) Without speculation (b) With speculation

2.1.4 Buffer Partitioning

The aspect of NoC design relevant to this thesis is buffer partitioning. The partitioning scheme of the input buffers can be classified as [32]:

- A single central memory across all the input ports and virtual channels. This memory automatically acts as a switch since the output port has to just read from the appropriate memory location. However, the memory read and write speed become a network bottleneck and deteriorates performance.
- A separate buffer for every input port. This requires a switch whose number of inputs is equal to the number of input ports.
- A separate buffer for every VC (virtual channel) per input port. This provides greatest flexibility since it allows a switch to read from two virtual channels in the same cycle. However, the overall buffer space is not utilized optimally since an idle virtual channel means wasted buffer space.

The work in this thesis assumes a separate buffer for every virtual channel for every port.

2.2 Emerging Non-Volatile Memories

Since several decades, SRAM and DRAM, which are volatile memory technologies are being used for caches and main memory. New non-volatile memory (NVM) technologies have a potential to change the design of memory systems significantly. One of the main characteristics of all the NVMs is their ability to store data even under no presence of power. As a result, they were mainly used as Read Only Memory (ROM). The emergence of NVMs like PCM, RRAM and MRAM has started to change the design trend. We will walk through the basic working of PCM and STT-MRAM below.

2.2.1 Phase Change Memory (PCM)

PCM is the most mature of the new memory technologies. The storage element consists of a chalcogenide: a phase change material and a resistor between two electrodes. A PCM storage element is shown in Figure 4.

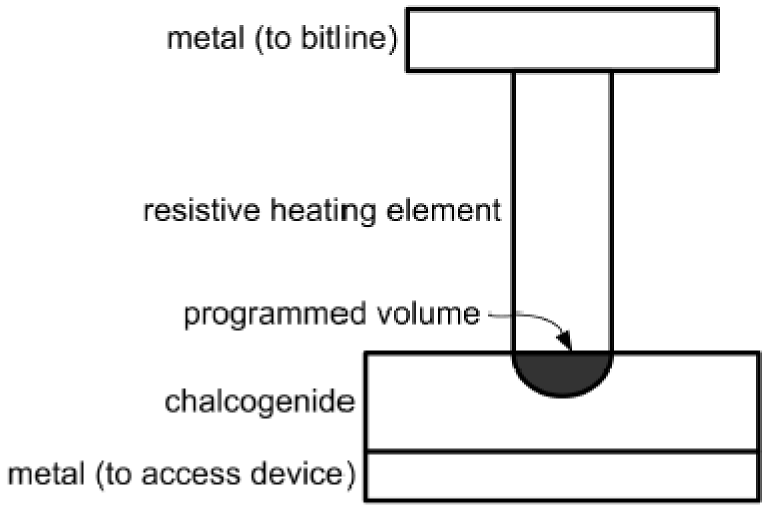


Figure 4. A PCM storage element

The phase change material has two states: amorphous and crystalline. In amorphous state, the material has very high resistance and in crystalline state, it has low resistance. These two states are also called RESET and SET respectively. The PCM is RESET by a short burst of high current and is SET by a long duration of moderate current which ramps down with time [12]. These two states of the PCM can be used to store logic 0 and 1. It is also possible to store more than 1 bit per cell, these are called Multi-Level Cells (MLC) owing to the great difference in resistance.

Since the SET duration is long, the write latency is determined by SET duration and write energy is determined by RESET current. All the NVMs suffer from long write latency and write energy compared to read. However, the biggest disadvantage of PCM is its endurance. The write endurance varies from 10^4 to 10^9 writes.

PCM is now being explored as an alternative to DRAM [12].

2.2.2 Spin-Torque Transfer Magnetic RAM (STT-MRAM)

STT-MRAM is a next generation memory technology 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 MgO tunnel barrier layer in the middle. Among them, the magnetization direction of the bottom layer is fixed. The spin of the electrons in the top layer is influenced by passing current through the fixed layer to polarize the current, and the current propagates through the free layer. Depending on the current, the spin polarity of the free layer changes either parallel or anti-parallel to that of the fixed layer. The parallel indicates a zero state, and the anti-parallel a one state.

Figure 5 depicts the parallel and anti-parallel states of an MTJ module. A single MTJ module is coupled with an NMOS transistor to form a basic memory cell of STT-MRAM called a 1T-1MTJ cell.

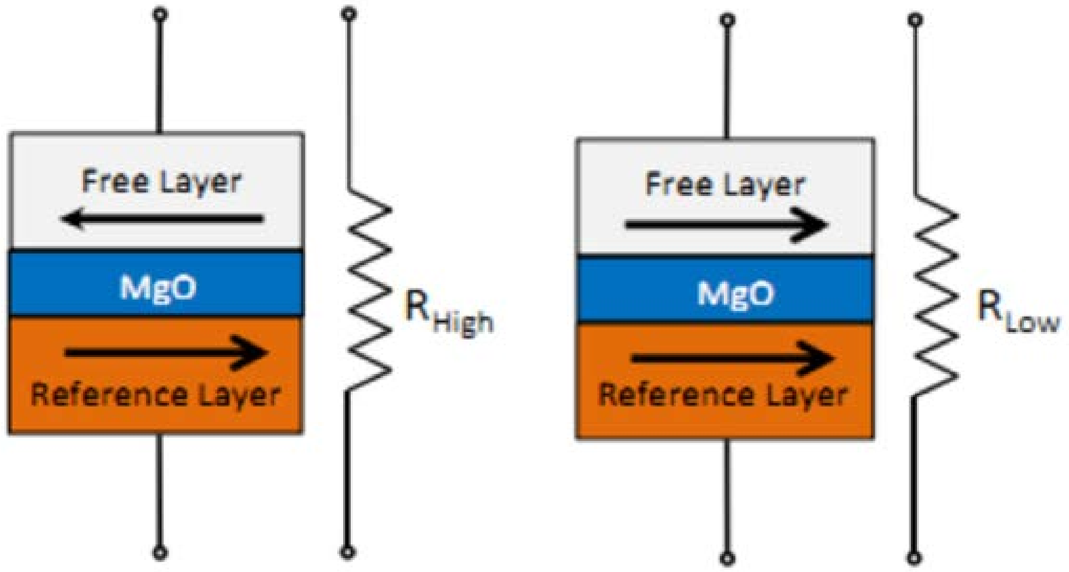


Figure 5. The two states of an MTJ module

2.2.3 Comparison of Different Memory Technologies

As shown in Table 1. STT-MRAM combines the speed of SRAM and density of DRAM. It also has low leakage power, however dynamic power is high. This is reduced by trading off with retention time.

eDRAM offers 3x density advantages compared to SRAM. The random write cycle time at 45nm technology comes to about 1.3-1.9ns. Our STT-MRAM memory cell offers 4x density advantages and the write time is 1ns.

Table 1. Comparison of SRAM v/s eDRAM v/s STT-MRAM

Technology	SRAM(6T)	eDRAM	STT-MRAM
Access time	fast	slow	fast
Density	low	high	high
Leakage	high	low	low
Refresh	no	yes	no
Destructive reads	no	yes	no

3. RELATED WORK

3.1. Utilizing NVMs in Processors and Memories

Jog et al. [9] attempted to achieve better write performance as well as energy consumption of L2 cache with STT-MRAM through adjusting data retention time of STT-MRAM. Similarly, Smullen et al. [23] reduced the write latency and dynamic energy of STT-MRAM by tailoring the retention time for on-chip caches. In [15], they integrated on-chip caches based on STT-MRAM in a 3D CMP environment and hided the long write latency by delaying cache accesses to busy STT-MRAM banks. Prior to that, Sun et al. [24] stacked MRAM-based L2 caches on top of CMPs and reduced overheads through read-preemptive write buffer and hybrid cache design using both SRAM and MRAM. Guo et al. [5] addressed various design issues of microprocessors using STTMRAM for power-efficient CMP systems. Most recently, Sun et al. [25] deployed STT-MRAM in L1 caches as well as L2 caches in CMP systems. The improved the write performance of STT-MRAM in L1 caches by adjusting the retention time of STT-MRAM in a ultra-low level, while providing a power efficient refreshing mechanism to prevent data corruption in L1 caches.

PCM also has been constantly explored to replace existing SRAM or DRAM-based memory systems. PCM shows higher density, however, it suffers from lower endurance compared to SRAM or STT-MRAM. These characteristics cause PCM to be adopted mainly for off-chip memories rather than on-chip caches. PCM-based main memory scheme were discussed in [31, 19, 12]. In [18], adaptive write cancellation and write pausing policies contributed to the reduction of energy and performance overheads. Zhou et al. [30] suggested a new memory scheduling scheme that allows Quality-of-Service (QoS) tuning through request preemption and row buffer utilization. Jiang et al. [8] improved the performance of multilevel cell (MLC) PCM through reducing the write iterations using extra error correction code (ECC). (With a scheme to reduce the storage overheads of the ECC.)

3.2. Buffer Management in Interconnection Networks

Efficient management of input buffers in on-chip interconnection networks is critical in improving the networking performance, especially in terms of network throughput. Buffer storage allocated per network channel is divided into multiple small queues, each of which is assigned to a VC. Virtual channel flow control [2] helps to reduce Head-of-Line blocking, thus improving the overall network throughput. Tamir and Frazier [26] proposed Dynamically Allocated, Multi-Queue (DAMQ) buffer that allows the dynamic partitioning of buffer space between the output ports to improve buffer efficiency. The dynamic virtual channel regulator (ViChaR) [16] also allows dynamic management of buffer resources and additional dynamic allocation of VCs per output port, which improved network throughput.

There have been attempts to utilize inter-router links as buffer storage to reduce buffer overheads. Kodi et al. [11] proposed the use of dual-function links, which utilizes inter-router link repeaters as buffers to reduce the power and area overheads while sacrificing marginal performance degradation. Michelogiannakis et al. [14] presented elastic buffers (EBs) to use the channel buffers in place of existing VCs, which prevents deadlock through duplicating physical channels.

Jang et al. [7] proposed a hybrid input buffer scheme utilizing both SRAM and STT-MRAM in input buffers of NoC routers to achieve better network throughput and power efficiency compared to purely SRAM-based counterparts. Since STT-MRAM shows high density and near-zero leakage power, they adopted STT-MRAM to increase network throughput while saving standby power. In addition, to overcome the long write latency and significant dynamic power consumption of STT-MRAM. Due to the inevitable use of SRAM, however, hybrid input buffers have limitation in maximizing the benefits of STT-MRAM in terms of total router power consumption including both dynamic and leakage power.

4. DESIGN CONSIDERATIONS

In this section, we describe important design issues of a STT-MRAM in terms of various data retention time, power consumption and area it takes up in on-chip interconnection networks, also present a potential packet dropping problem that could be caused when the total duration of a flit buffered in STT-MRAM exceeds the given retention time of the buffer.

4.1 Modeling STT-MRAM

The data retention time, T_{ret}, of an MTJ cell is defined as follows [20].

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

Δ is the thermal factor that estimates the thermal stability of an MTJ module. The thermal factor depends on the in plane anisotropy field (H_k), the saturation magnetization (M_s), working temperature (T) and the volume of an MTJ cell (V). In a precessional switching mode where an MTJ switching time (T_s) is short (<3ns), the required current density, $J_c(T_s)$, is determined as follows.

$$J_c(T_s) \propto J_{c0} + \frac{C}{T_s}$$

J_{c0} is a process-dependent switching threshold current density that depends on M_s and H_k [28]. 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 as well, which reduces H_k and M_s, and eventually decreases J_{c0}.

Therefore, with smaller J_{c0}, we can achieve shorter switching time with the same current density. 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.

It is impractical to reduce the write latency of STT-MRAM as fast as SRAM because it involves huge sacrifice of the retention time, thereby possibly resulting in significant amount of packets dropped in input buffers.

Reducing the retention time up to 100ns allows us to obtain smaller J_{c0} by shrinking H_k and M_s. Through this, 2 cycles of write latency, which corresponds to 1ns in 2GHz clock frequency, is achieved with 280µA of switching current, which can be provided by 31.3F^2 of STT-MRAM cell size. These results are based on the simulation with the PTM model [1] under 32nm technology. Throughout this paper, unless otherwise stated, writing a flit to STT-MRAM buffer takes 2 cycles which corresponds to 1ns in 2GHz clock frequency.

Dynamic write energy consumption of STT-MRAM cell is calculated by:

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

where V is power supply voltage and R_{MTJ} is the resistance of an MTJ cell [29]. Since the write energy is proportional to the switching time, T_s, the energy consumption reduces as the write latency decreases, which supports the energy efficiency of our STT-MRAM model with less than 1ns of latency.

The STT-MRAM cell area is mostly determined by the NMOS transistor size [5] since the MTJ cell is much smaller compared with the transistor. Increasing the transistor size causes a large voltage drop to the MTJ cell and guarantees enough switching current. We model the STT-MRAM cell area based on the model in [9] as follows.

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

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

4.2 Packet Dropping Problem

As described earlier, the STT-MRAM cell can be optimally tailored for NoC routers to maximize the network performance, while minimizing power consumption associated with write operations. Its data retention time, however, may not be always sufficiently long enough to maintain the validity of the oldest packets staying inside the buffer of NoC router.

Regarding this, to observe such a potential packet dropping problem, we measured the maximum number of cycles of a flit staying in the buffer. Table 2 shows the result with different injection rates ranging from 0.1 to 0.7 under 8 by 8 mesh network.

Table 2. Maximum cycles of a flit staying in a buffer

Injection Rates	0.1	0.2	0.3	0.4	0.5	0.6	0.7
Max Cycles	30	48	99	205	288	327	376

In [7], Jang *et al.* assumes 10ms retention time of STT-MRAM, which corresponds to 20 million cycles in 2GHz clock frequency. It is an enough range to maintain the integrity of flits stored in a STT-MRAM buffer, thus no packets are dropped in NoC. If, however, we reduce the retention time further, the possibility of packets dropped in STT-MRAM buffer is increasing accordingly.

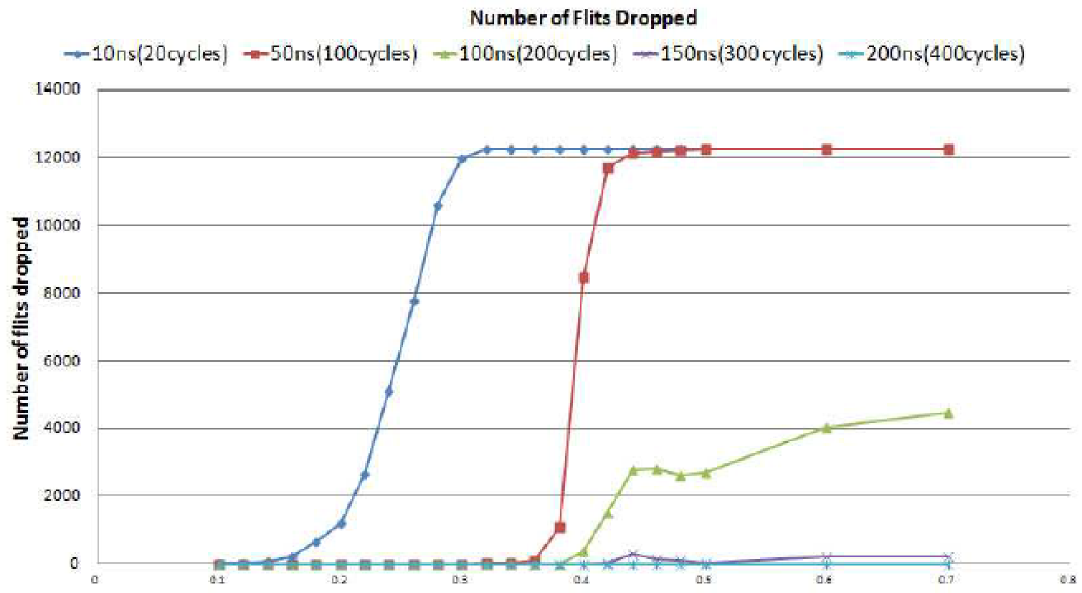


Figure 6. Retention time v/s dropped flits

Figure 6 shows the number of flits dropped in STT-MRAM buffer using different retention times ranging from 10ns to 200ns under 8 by 8 mesh network. The graph shows that under the same injection rate e.g. 0.5, as the retention time decreases from 200ns to 10ns, the number of flits dropped gets increasing, and no flits are dropped when retention time is greater than 200ns, which corresponds to 400 cycles in 2GHz clock frequency. Similarly, under the same retention time e.g. 100ns, as the injection rate increases from 0.1 to 0.7, the number of flits dropped is also increasing. This is because the congested network in a high injection rate leads flits to stay longer inside the buffer of on-chip routers than the lightly loaded network in a low injection rate.

Thus, in order to make the best use of the STT-MRAM, a timely and power-efficient buffer refreshing scheme should be provided to avoid such packet dropping problem. In Section 5.3, the details of such new refreshing schemes will be provided.

5. ON-CHIP ROUTER ARCHITECTURE WITH STT-MRAM

In this section, we describe a generic on-chip router architecture and its buffer structure and present our purely STT-MRAM based router architecture which enables writing data into buffers every cycle without incurring any additional delay through a dual ported and banked buffer organization, while preventing packet dropping problem.

5.1 Generic Baseline Router Architecture

The generic NoC router architecture is depicted in Figure 7. It is based on the state-of-the-art speculative router architecture [17]. Each arriving flit goes through 2 pipeline stages in the router: routing computation (RC), VC allocation (VA) and switch arbitration (SA) at the first cycle, and switch traversal (ST) at the second cycle. A lookahead routing scheme [4] is adopted, which 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 [3]. Credit-based VC flow control [2] is adopted to provide the back-pressure from downstream to upstream routers, thus controlling flit transmission rate to prevent packet loss due to buffer overflow.

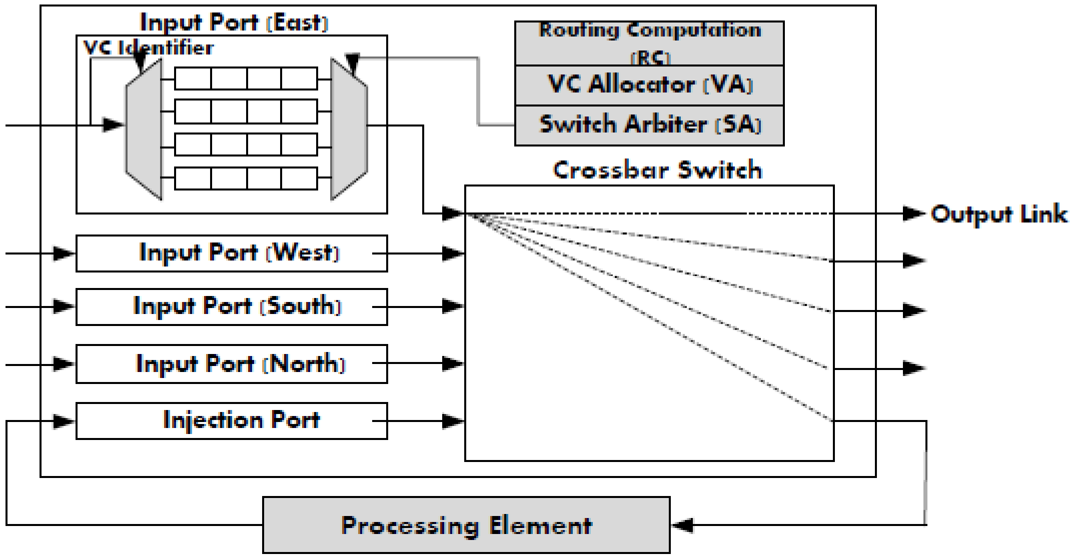


Figure 7. Generic baseline router architecture

Due to the limited area and power resources and ultra-low latency requirements, on-chip routers rely on very simple buffer structure. VC-based NoC routers consist of a number of FIFO buffers per input port where each FIFO corresponds to a VC. Each input port has *v* VCs, each of which has a *k*-flit FIFO buffer. Current on-chip routers have small buffers to minimize area overheads, thus *v* and *k* are much smaller than in macro networks. The necessity for ultra-low latency leads to a parallel FIFO buffer design. Contrary to a serial FIFO implementation, the parallel structure eliminates unnecessary intermediate processes for a flit to traverse all buffer entries until it leaves the buffer [27]. This fine-grained control requires more complex logic, which manages read and write pointers to keep the FIFO order. The read and write pointers in the parallel FIFO registers control an input de-multiplexer and an output multiplexer. The write pointer points to the tail of the queue, and the read pointer points to the head of the queue. For a read operation, the flit pointed by the head is selected and transmitted to a crossbar input port. Similarly, write operation leads the incoming flit to be written to the location pointed by the tail pointer. The pointers are promptly updated after each read or write operation. After a read operation, once the head is overlapped with the tail, the buffer becomes empty. After a write operation, likewise, if the tail moves to the same position pointed by the head, the buffer is full.

5.2 Pure STT-MRAM On-Chip Router Architecture

5.2.1 Router Design Consideration

In a highly congested network, flits are arriving at the input buffer of downstream router at every clock cycle. In this case, for conventional pure SRAM router, incoming flits are written to appropriate buffers available at each clock cycle without any delay since it usually takes less than one cycle storing a flit to the buffer. When we replace the SRAM with STT-MRAM, however, only one flit can be written to the buffer at every 2 cycles, which can lead to the dropping of subsequent incoming flits.

In a lightly loaded network, contrarily, flits may not be dropped, but due to the multiple write cycles taken, the internal router pipeline stages are stalled, thus ultimately increasing the overall network latency by leading flits to reach to its destination late.

Either way, therefore, hiding this multiple cycle write latencies is of paramount importance to guarantee maximum network throughput without incurring any performance degradation.

5.2.2 Dual Bank STT-MRAM Buffer

To hide the 2 cycles of write delay of STT-MRAM input buffer, we divide each VC buffer into two banks with interleaved addresses. For simplicity, we refer to them separately as **Odd** and **Even** banks. In this, every odd numbered flits in STT-MRAM input buffer are sent to an Odd bank, and likewise, even numbered flits to even bank. Figure 8, for instance, shows a black box which has one input port, and two splitted output ports. In the figure, every odd numbered flits in input port are sent to odd output port, and likewise, even numbered flits to even output port. The rationale behind this is that every odd/even flit arrives at the input buffer every 2 cycles, so by making odd/even bank hold its incoming flits for 2 cycles, we can make input buffers store flits every clock cycle.

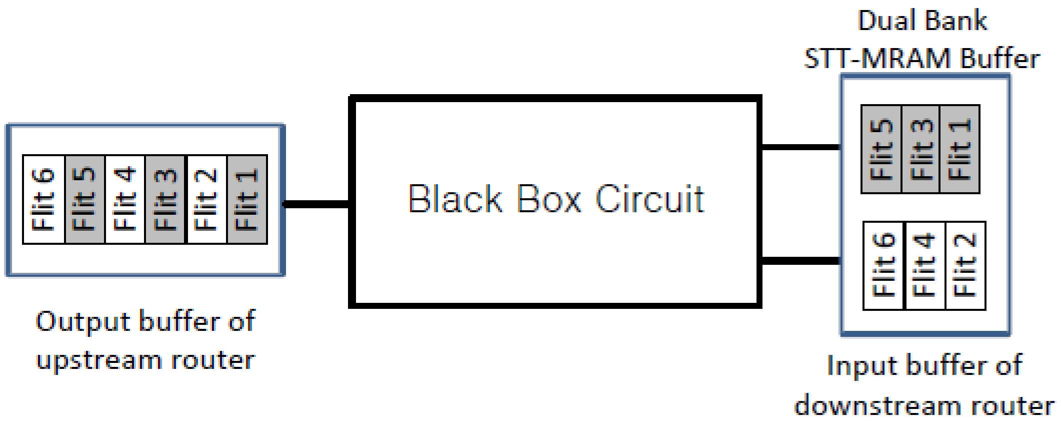


Figure 8. Black box circuit

Figure 9 shows the internal structure of the black box explained above. A clock controlled latch L1 at the input port of the router is used to pipeline consecutive flits. For every clock cycle, only one flit is allowed to pass through, while the next incoming flit is latched in L1.

Muxes are simple 2:1 multiplexers that isolate consecutive flits and decouple their data paths in the router. These muxes are controlled by a common select signal which changes once every clock cycle, or in other terms, is itself a clock of half the frequency of the interconnect clock. It is noted that inputs of the mux are connected to the input and output of L1. Since L1 is synchronized with the interconnect clock, for every arriving flit, both inputs change at every cycle. These changes are carefully orchestrated in such a way that the output of both the muxes are valid for at-least 2 cycles. Since consecutive arriving flits can have different VCs, separate VC decoders are employed for each MUX.

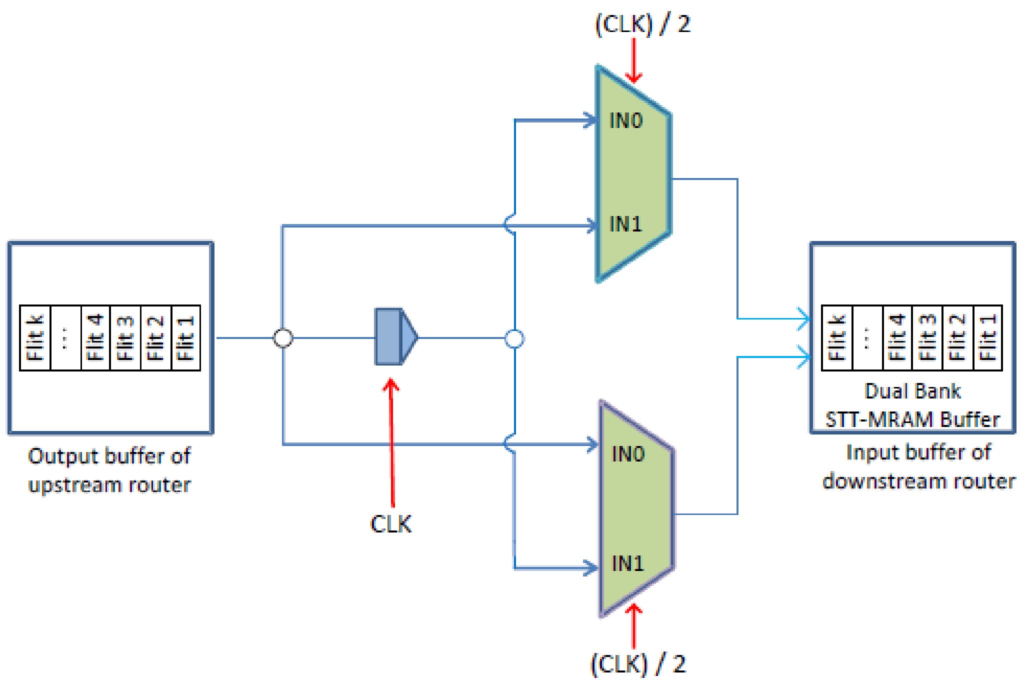


Figure 9. Inside of black box

5.2.2.1 Dual Banked STT-MRAM Buffer Working Example

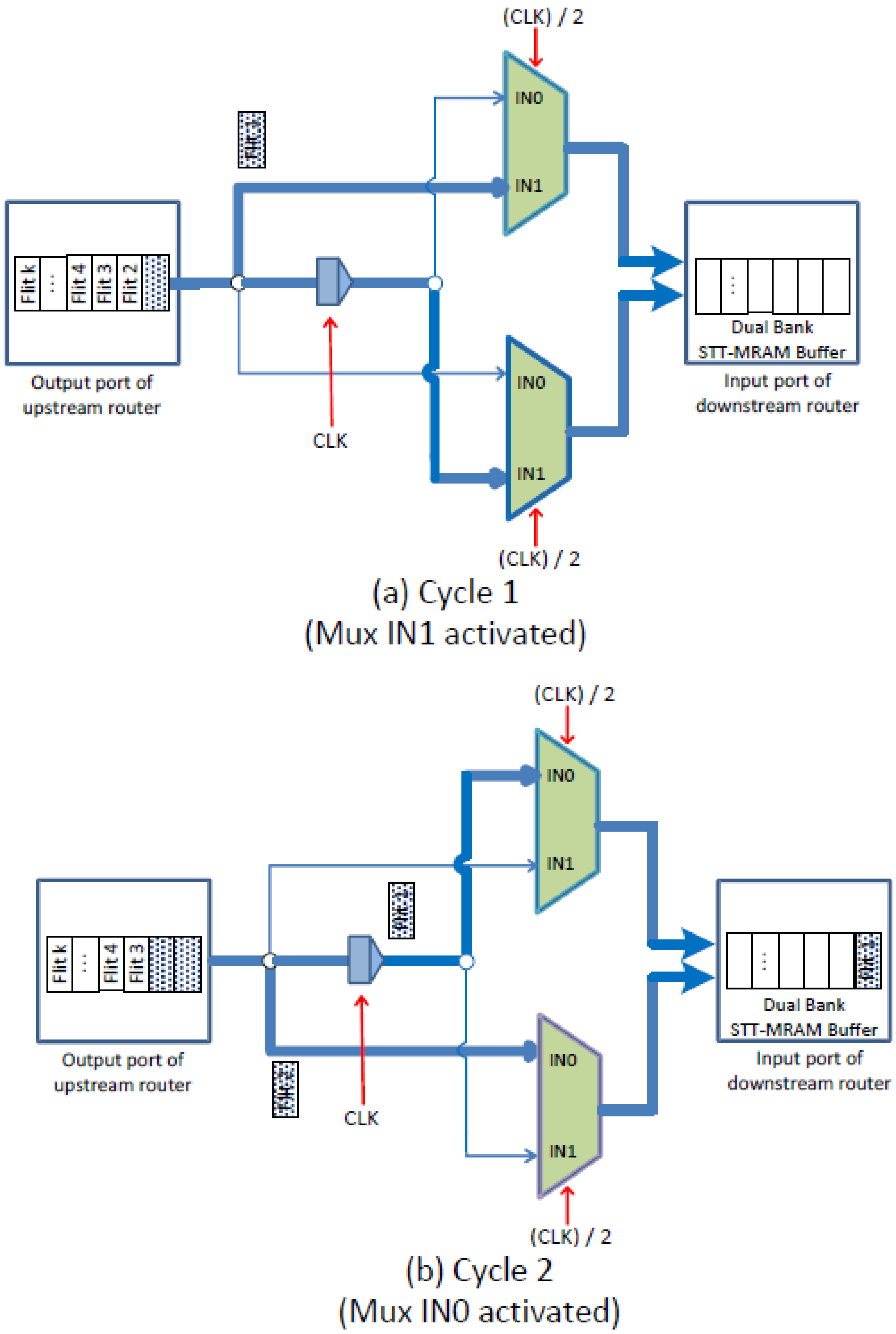
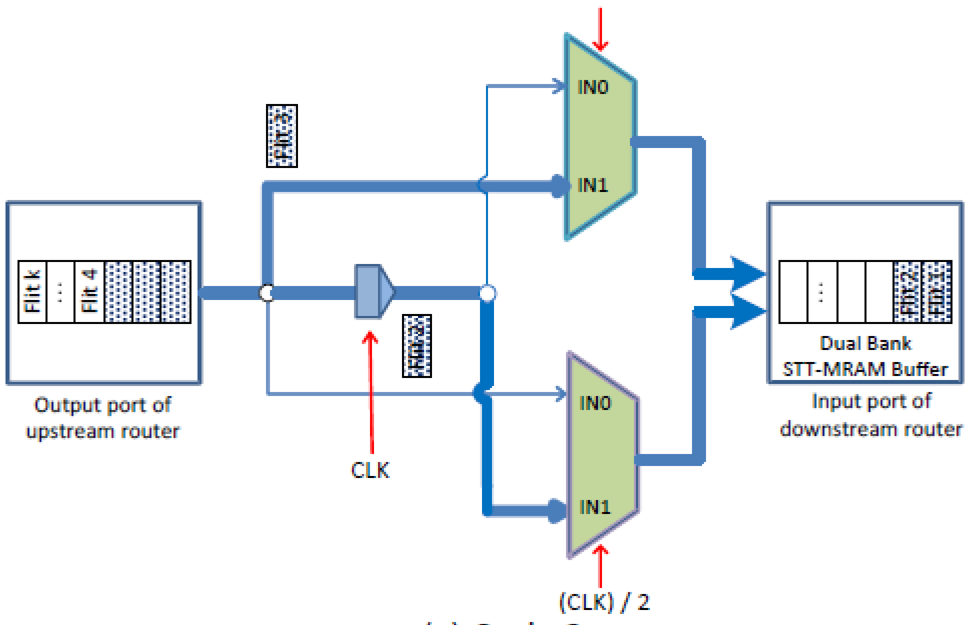


Figure 10. STT-MRAM dual bank example



(c) Cycle 3
(Mux IN1 activated)

Figure 10. Continued

Figure 10 shows an example of the data flow of 4 incoming flits at consecutive clock cycles. Initially the control of Mux0 and Mux1 is assumed to be set to 0, and all VCs are empty.

- Cycle 1: This is the first write cycle for Flit1. Flit1 is latched at L1, and at the same time, the IN1 of top Mux and IN0 of bottom Mux are now set to Flit1. The input of VCD1 also contains Flit1 since the control signal of Mux is 0. The input of VCD0 is still invalid since there is no data at IN0 of top Mux. The bank selector logic sets the bank-bit of all VCs to be 0 or even banked because all the VCs are empty

- Cycle 2: This is the second write cycle for Flit1 and its write completes by the end of this cycle. Flit2 is latched at L1, and similar to Cycle 1, IN1 of top Mux and IN0 of bottom Mux has Flit2. Flit1 is emptied from the latch and populates IN0 of top Mux and IN1 of bottom Mux. The control of both Muxes has changed to 1, and this implies that the output of top Mux is Flit2, and that of bottom Mux is still Flit1. It is safe to assume that the interconnect clock period is long enough to satisfy the setup and hold constraints of a simple CMOS MUX.

  The input of VCD0 is set to Flit2. If Flit2 has the same VC-ID as Flit1, its bank selector logic detects the ongoing write in bank0 and sets the bank-bit for VCD0 to be 1. With a dual bank buffer, Flit2 is being written to the odd bank in this cycle. If Flit2 has a different VC-ID from Flit1, the bank-bit is set to the appropriate bank of that VC. In both cases, this is the first write cycle for Flit2.
- Cycle 3: This is the second write cycle of Flit2 and the first for Flit3. The select signal for M0 and M1 switch back to 0. The bank selector logic orchestrates the appropriate banks for each write.

To sum up, a tabular representation of consecutive flits all destined to the same VC is shown in Table 3.

Table 3. Tabular representation of consecutive flits

Clock Cycle	L1	MUX Sel	M0		M1		VCD0		VCD1		Write progress
			IN0	IN1	IN0	IN1	Input	Bank_bit	Input	Bank_bit	
CC0	f0	0	-	f0	f0	-	-	-	f0	0	f0 begins
CC1	f1	1	f0	f1	f1	f0	f1	1	f0		f1 begins, f0 completes
CC2	f2	0	f1	f2	f2	f1	f1		f2	0	f2 begins, f1 completes
CC3	f3	1	f2	f3	f3	f2	f3	1	f2		f3 begins, f2 completes

5.2.3 Buffer Read/Write Logic and Dual Bank Selector

The current read/write logic of a conventional SRAM input buffer maintains a head and tail pointer which is updated at every clock cycle. For every read from the buffer, the head pointer is increased. For every write, likewise, the tail pointer is increased. We intend to pipeline the writes to separate banks of the same buffer. Since the banks are interleaved and preserve a contiguous address space, the head and tail pointer of the STT-MRAM buffers still get updated every cycle. As a result, there are no architectural changes required for the read/write logic of the pure STT-MRAM buffer.

For buffer writes, a dedicated and independent write port is provided for each bank. This ensures that there are no structural hazards for flits arriving at consecutive cycles. Essentially, if the second flit arrives while the first one is still being written, it is directed to another bank to start its writing process without being blocked. Since the bank addresses are interleaved and present a logically contiguous address space, the second flit is always written at its appropriate buffer location, ensuring in-order writes.

For buffer reads, a single read port is sufficient because read latencies of a single STT-MRAM is similar to that of an SRAM. Once per cycle writes are ensured for the buffer, the read logic does not need any adaptation from the SRAM design.

The bank selector unit decides which bank of a VC the flit should be destined to. Each VC decoder has its own bank selector unit that computes the bank-bit for all VCs.

The VC-ID field of the MUX outputs specifies which bank bit is used. As is described in the flowchart (Figure 11), the bank selector logic works in the following way:

1. Both the write ports of a VC are polled to check if it any of them is busy. If one of the write ports is busy, the bank-bit for that VC is set to the non-busy bank.
2. If both write ports are idle, the last bit of the tail pointer is checked. The bank-bit for that VC is then set based on this value. If the last bit is 1, bank bit is set to 0 or the even bank, else it is set to point towards the odd bank.
3. The output of the bank-bit for each VC decoder remains valid for 2 cycles before the write ports are polled again. This ensures that flits arriving at consecutive or nonconsecutive clock cycles with different or same destination VC are contiguously placed inside their respective VC buffers without any empty slots.

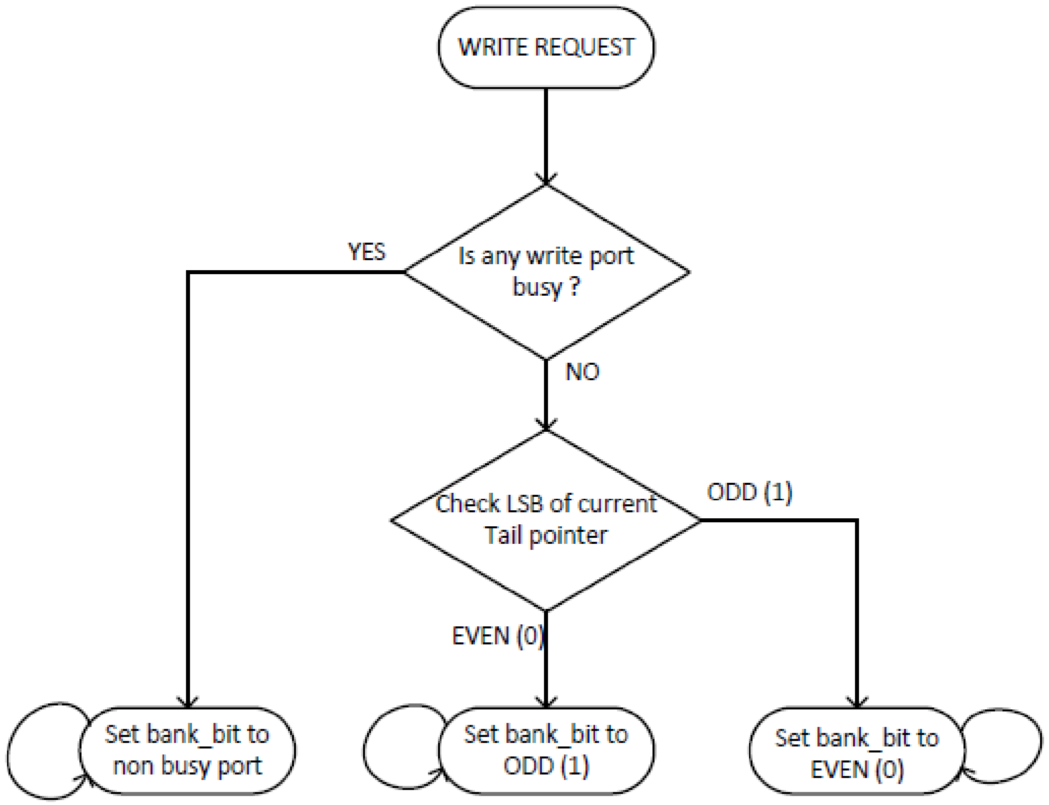


Figure 11. Bank selector logic flowchart

The bank selector logic also keeps track of the head and the tail pointer for each VC buffer. If and when a flit arrives towards a VC whose head and tail pointers match, the bank selector can enable buffer bypassing to save a redundant buffer write. This is implemented via a 2:1 MUX for each VC.

5.2.4 Router Microarchitecture

The detailed architecture of a purely STT-MRAM router to enable single write per cycle is shown in Figure 12.

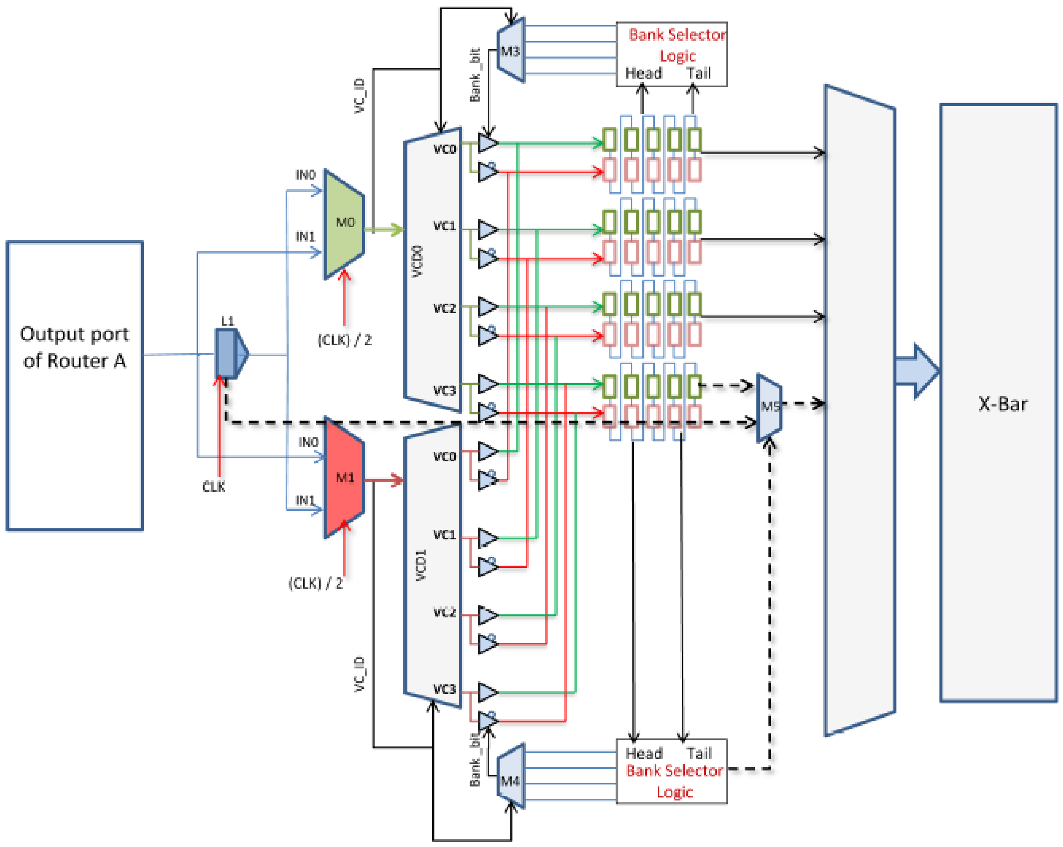


Figure 12. Pure STT-MRAM router microarchitecture

For a generic router design with 4 VCs, two 1:4 decoders are needed. These are labeled as VCD0 and VCD1. Since the virtual channel ID of a flit now needs to be valid for two cycles to ensure a successful consecutive write, the control signal of the decoders are extracted from the VC-ID field of their respective MUXes.

A series of simple tri–state buffers are interfaced with the STT-MRAM buffers to provide each VC decoder output access to both banks of their respective VC. A tri-state buffer acts as a wire if its input is enabled and high impedance otherwise. The output of the tri-state buffers are directly connected with the write ports of the banks they are interfaced to. Each VC decoder output has access to both even and odd banks via these tri-states.

The bank selector logic described in Section 5.2.3 controls the tri-state buffers and governs the bank where each flit is destined to. It polls the current buffer occupancy of each VC and sets the **bank-bit** to either 0(even) or 1(odd) based on the next available bank. The VC-ID signal used by the VC decoders is also used to control the selection of the bank-bit of the appropriate VC. The same VC buffer is shared between VCD0 and VCD1, but each has its own bank-bit. It is noted that each VCD also has its own bank selector logic.

5.2.5 Buffer Bypassing

Bypassing the input buffer is a common optimization for performance. Bypassing the STT-MRAM buffer, a flit can be sent directly to the switch arbitration stage whenever the VC is empty. This reduces the pipeline depth by one stage and impacts performance favorably. The flit can be bypassed immediately after it is available in the latch L1. The bank selector logic maintains the head and tail pointer and can easily detect if the VC buffer is empty. This is used by the MUX M5 (only shown for one VC) to select the flit from either bypass path or VC buffer. If bypass succeeds, there is no need to read from the buffer and the flit can be dropped, thus saving read energies. However if it fails due to unavailability of the switch, the write process still continues as before without any penalty. This technique is more effective for low congestion traffic where bypasses are common.

5.3 Prevention of Packet Dropping

As is described earlier in Section 4.2, packet is dropped when the retention time of STT-MRAM is less than the duration of packets staying in STT-MRAM buffer. Under the same retention time, as packet injection rate goes up and network gets congested, more packets tend to be dropped. To prevent this problem, we can either adjust the retention time of STT-MRAM or utilize schemes to refresh STT-MRAM buffer periodically before the corruption of packets.

In this subsection, we describe two different refreshing schemes, simple and lazy STT-MRAM buffer refreshing, and compare the power-efficiency of the two refreshing schemes to a conventional DRAM-style periodic refreshing. In addition, for effective and power-efficient prevention of packet dropping, age-based switch arbitration is applied to each refreshing scheme suggested.

5.3.1 Age-Based Switch Arbitration

In the generic baseline router architecture, we assume round-robin switch arbitration in both VA and SA stage. It does not assign any priority in the arbitration stage, guarantees fairness and no starvation among input ports, and provides implementation simplicity because no comparisons are needed among the queues. The worst case wait time is proportional to one less than the number of requestors.

To minimize the number of packets dropped in on-chip networks, we observe that the use priority-based switch arbitration is more effective. The arbitration is made deterministically based on the relative age of the packets queued in the input buffer and the oldest packet is chosen over the others.

To quantitatively explore the effect of age-based switch arbitration on packet dropping, we measured the maximum cycles of a flit staying in the buffer. Table 4 shows these results with different injection rates ranging from 0.1 to 0.7 on 8-by-8 mesh network. The results of the round-robin switch arbitration from Table 2 are given for reference.

Table 4. Maximum cycles of a flit staying in a buffer with age-based SA

Injection Rates	0.1	0.2	0.3	0.4	0.5	0.6	0.7
Max Cycles (RR)	30	48	99	205	288	327	376
Max Cycles (Age-based)	14	24	48	146	170	172	189

Compared to the round-robin arbitration, maximum cycles of a flits staying in a buffer have almost halved in value. Figure 13 shows the difference in degree of packet dropping compared to round-robin based switch arbitration. Age-based switch arbitration is very effective in reducing the number of packets dropped, even though it does not completely prevent the packet dropping.

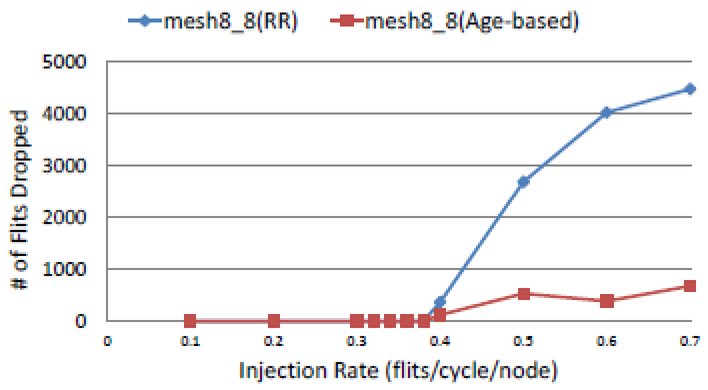


Figure 13. RR v/s age-based dropped flits

5.3.2 Simple Refreshing Scheme

Typically, refresh logic is provided in a DRAM controller which automates the periodic refresh, that is no software or other hardware has to perform it. In [23], a simple DRAM-style refreshing scheme was presented.

It reads the contents of a cache line and writes it back after performing error correction. The periodic refresh interval is calculated to guarantee that every line will be refreshed within the retention time interval. This scheme refreshes all cache blocks in sequence regardless of its data contents. However, this scheme introduces many unnecessary refreshing operations whose elimination will significantly improve performance and save energy.

To prevent packet loss introduced by relaxed non-volatility of STT-MRAM buffers in simple refresh scheme, we assign one counter to each virtual channel to monitor its data retention status. We define a refresh threshold, N, which signifies lifespan of a packet remaining before a possible packet drop. It is used to determine if the flits should be refreshed or not at a certain point of time. Our simple refreshing scheme simply checks the age of a flit queued in the header of each buffer. If it is less than N, then the flits are refreshed.

The threshold value is obtained by experiments which are designed to observe the maximum number of flits need to be refreshed at a certain clock cycle. As the refresh threshold gets smaller, the less refresh operation gets performed. The maximum value across the network is around more than 100 flits per cycle per router under 8 by 8 mesh, uniform random workload.

5.3.3 Lazy Refreshing Scheme

Lazy refreshing scheme counts the number of flits aged less than the refresh-threshold, which is similar to that of simple refreshing scheme, but it considers the ratio of the flits in STT-MRAM buffer to the total number of flits residing in the buffer. Once it exceeds a given refresh ratio, every flit in the buffer is refreshed.

To implement the lazy refresh scheme, counters are added to keep track of the flits in the STT-MRAM buffer and trigger the refresh adaptively. In this way, we can save total refresh power compared to both the DRAM-style conventional refreshing, and simple refreshing scheme above.

6. PERFORMANCE EVALUATION

6.1 System Configuration

A cycle-accurate NoC simulator [13] is used to conduct the detailed evaluation of the proposed scheme. It implements the pipelined router architecture with VCs, a VC arbiter, a switch arbiter and a crossbar. Under the 32nm process technology, unless specified, all simulations are performed in an 8x8 network having 32 out-of-order processors and 32 L2 cache banks on a single chip. We also perform experiments to test the efficacy of our scheme on network with different sizes. The network is equipped with 2-stage speculative routers with look ahead routing [4]. The router has a set of *v* VCs per input port. Each VC contains a *k*-flit buffer with 16B flit size. In our evaluation, we assume that *v* is 4, and *k* may vary with different buffer configurations. A dimension order routing algorithm, XY is used with wormhole switching flow control [22].

A variety of synthetic workloads [21] are used to measure the effectiveness of the STT-MRAM based on-chip router: uniform random **(UR)**, bit com plement **(BC)** and nearest neighbor **(NN)**. We also use different topologies to evaluate our design. Finally, we evaluate the efficacy of various refresh schemes in terms of refresh cycles needed.

6.2 Performance Analysis With Synthetic Workloads and Benchmarks

Performance of NoC is measured by latency and throughput. Latency is the number of cycles taken by a packet to reach the destination. Lowering the latency also increases the throughput since the packet flits take less time to reach the destination.

Figure 14, 15, 16 shows the performance analysis with BC, NN and TP respectively. Area of all the designs is kept the same for fair comparison. SRAM buffers with 6 flit buffers/vc/port, a hybrid design with 3 SRAM and 12 STT-MRAM flit buffers/vc/port, pure STT-MRAM design with 18 STT-MRAM flit buffers/vc/port all have about the same area. STT18 shows throughput improvement by 15% on average while hybrid S3_STT12 shows throughput improvement by 13% compared to SRAM6. These results can easily be explained by the fact that STT18 has higher buffer space than any other design. The effective write latency into the buffers is also same as SRAM and hybrid-design because of micro architectural changes. These buffer changes allow STT18 to delay network saturation and handle higher load.

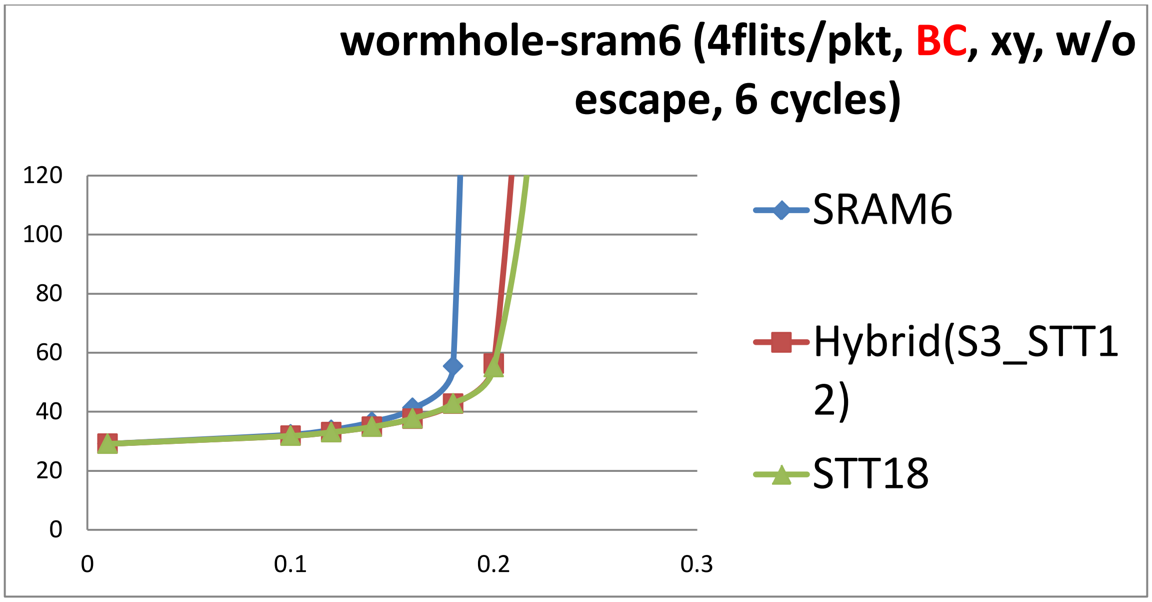


Figure 14. Performance analysis with synthetic workload: BC

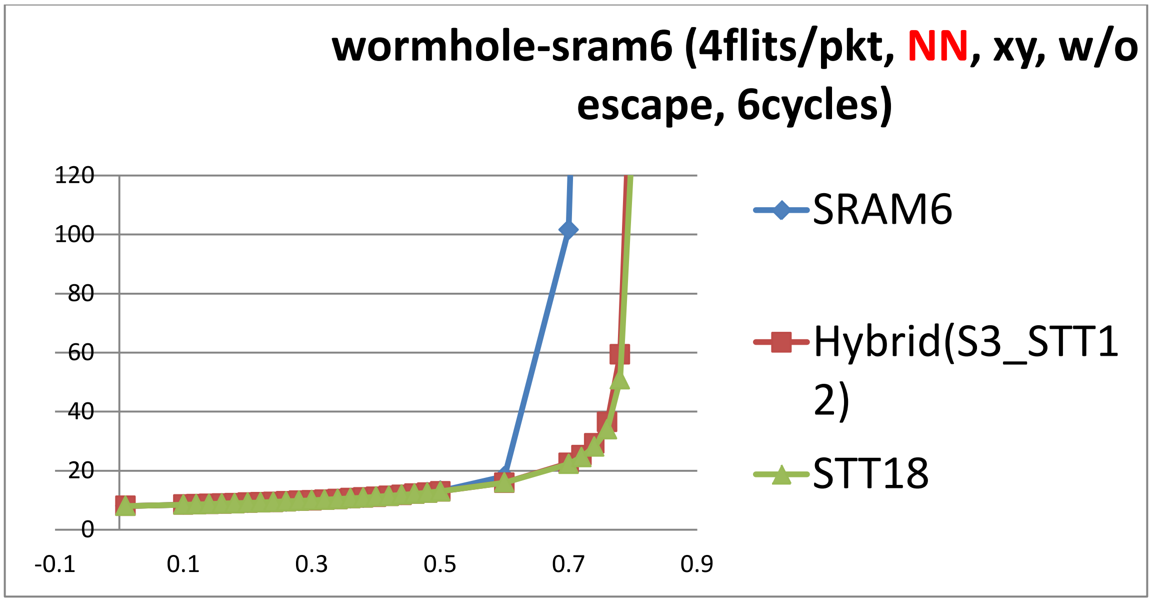


Figure 15. Performance analysis with synthetic workload: NN

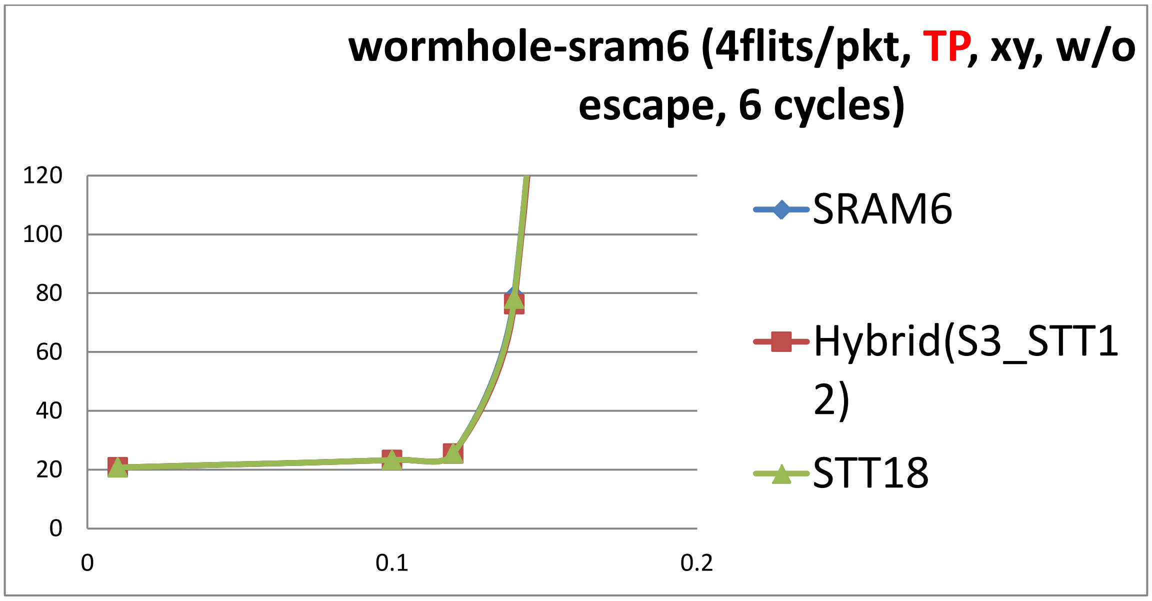


Figure 16. Performance analysis with synthetic workload: TP

Figure 17, 18, 19 shows the performance analysis with Cmesh, torus and flattened butterfly topologies. Compared to SRAM, the overall throughput is increased by 20.33%, 24.61% and 9.35% in Cmesh, torus and flattened butterfly topologies respectively compared to SRAM6. There is marginal difference in performance between hybrid and pure STT-MRAM designs.

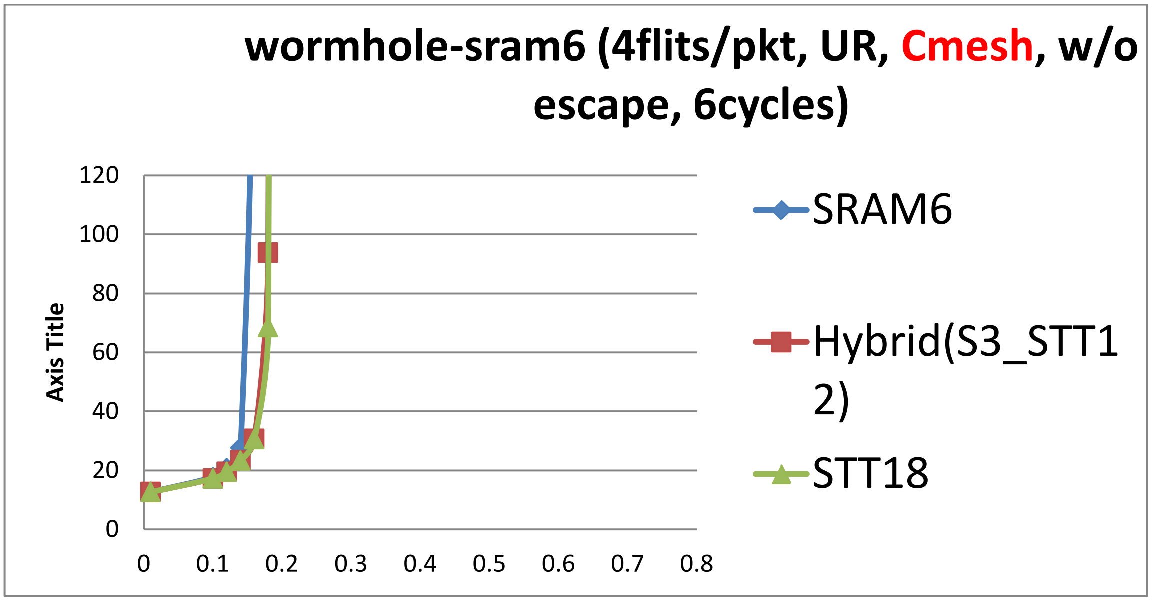


Figure 17. Performance analysis with Cmesh topology

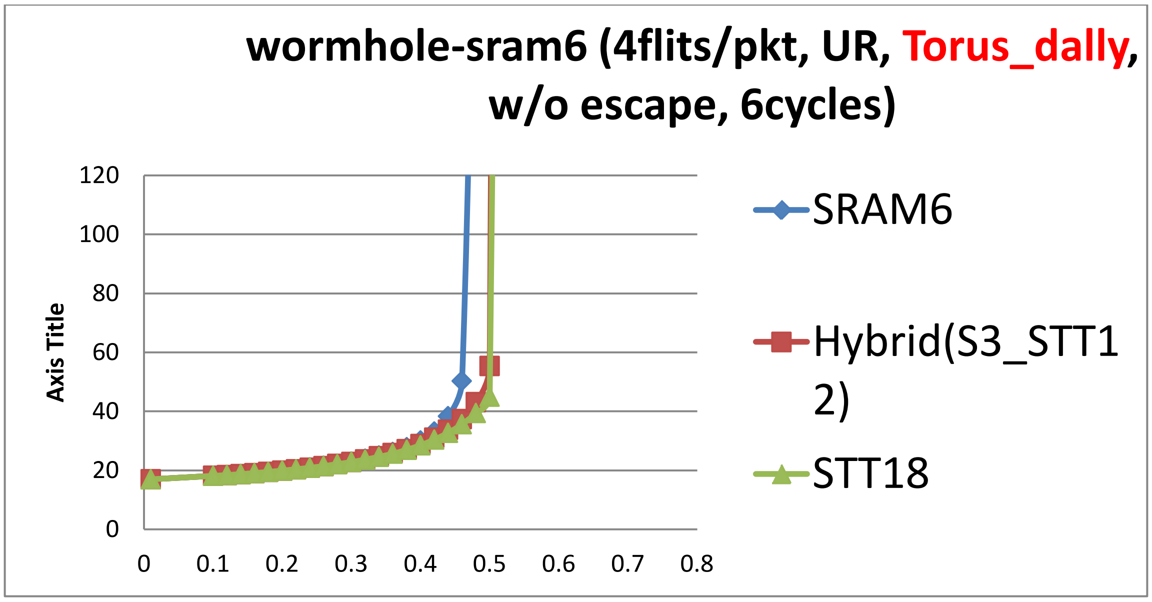


Figure 18. Performance analysis with torus topology

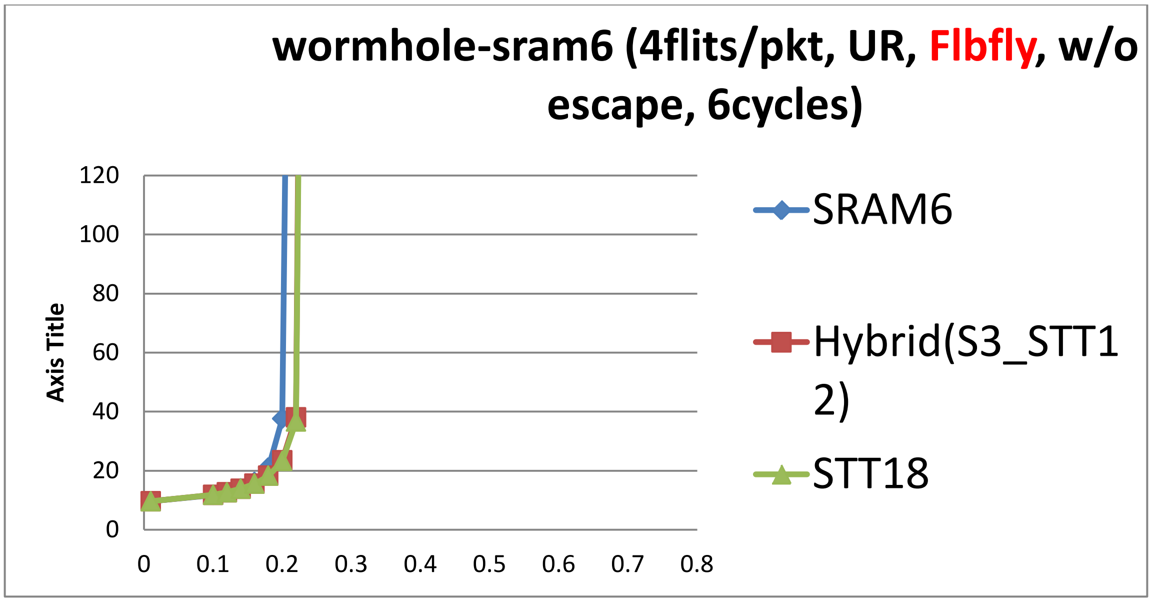


Figure 19. Performance analysis with flattened butterfly topology

Figure 20, 21, 22 shows the performance analysis with Uniform Random traffic and different network sizes. Compared to SRAM, the overall throughput is increased by 14%, 18% and 13% in 8x8, 12x12 and 16x16 mesh sizes respectively.

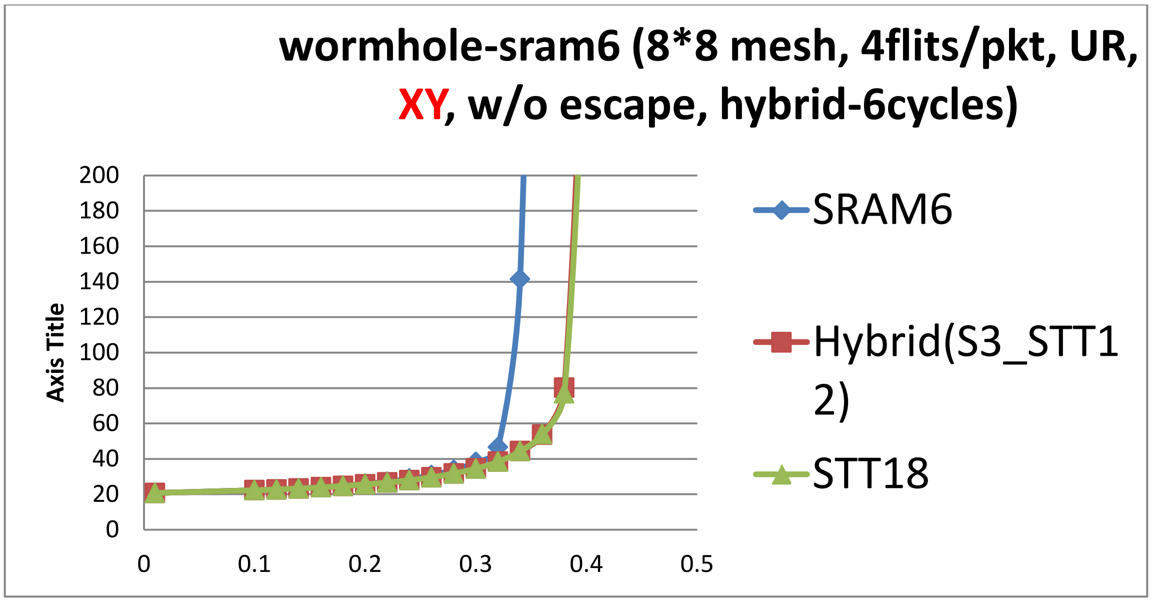


Figure 20. Performance analysis with UR, 8x8 mesh

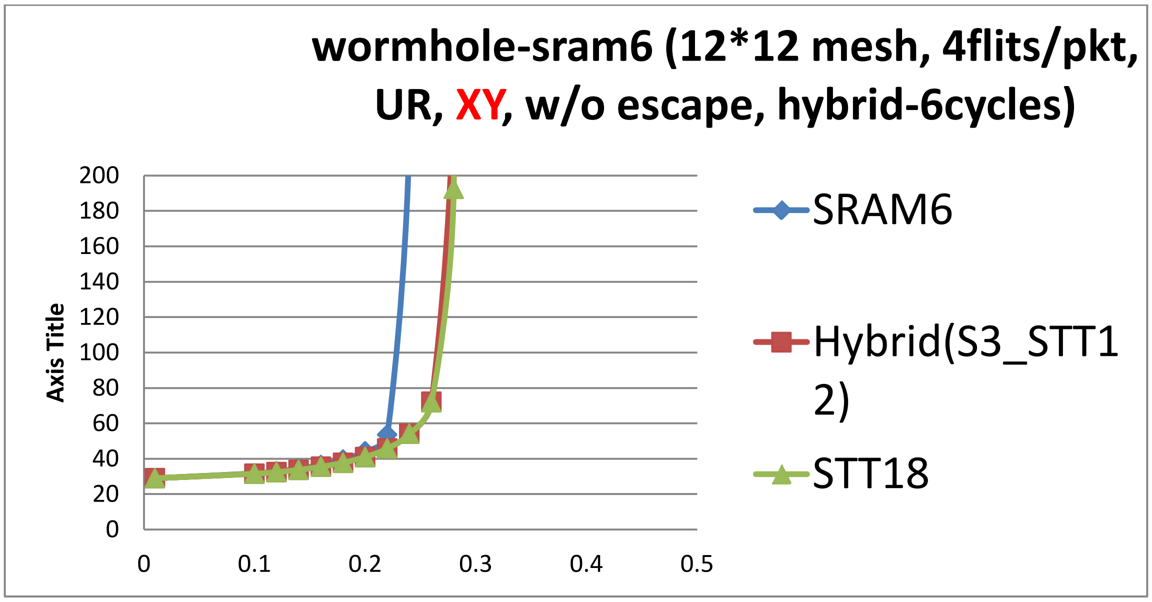


Figure 21. Performance analysis with UR, 12x12 mesh

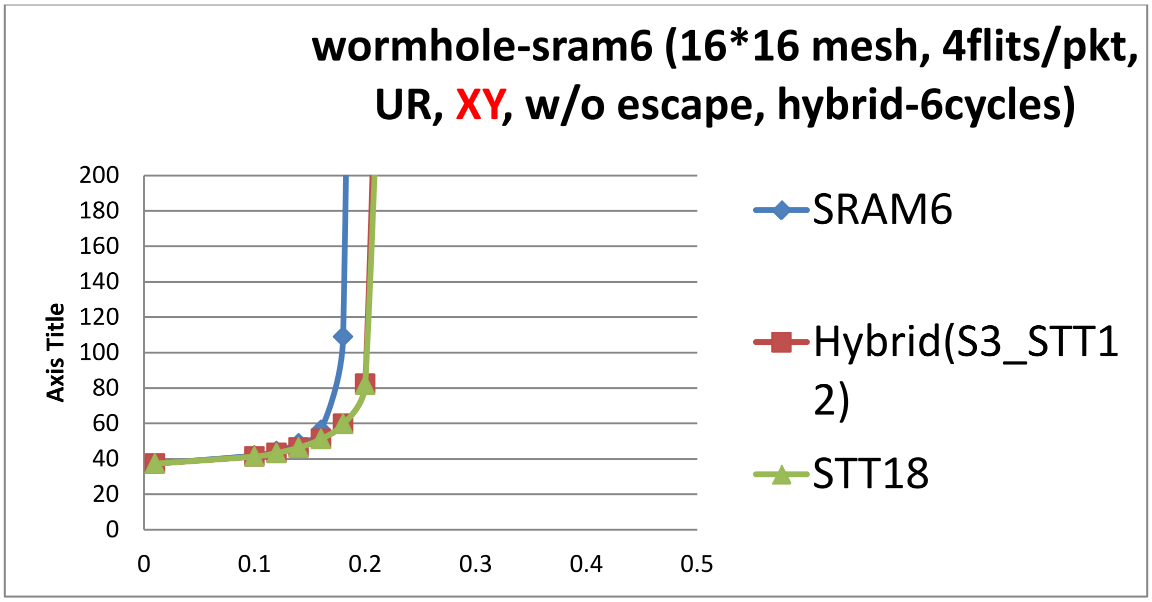


Figure 22. Performance analysis with UR, 16x16 mesh

Figure 23 shows the number of flits refreshed in simple and lazy refreshing schemes. Lazy refreshing scheme reduces the number of flits refreshed and thus saves power.

Figure 24 shows the number of flits dropped. There is no loss of flits in simple refreshing scheme. Under a very aggressive lazy refreshing scheme (75% ratio), flits are dropped and thus, not practical.

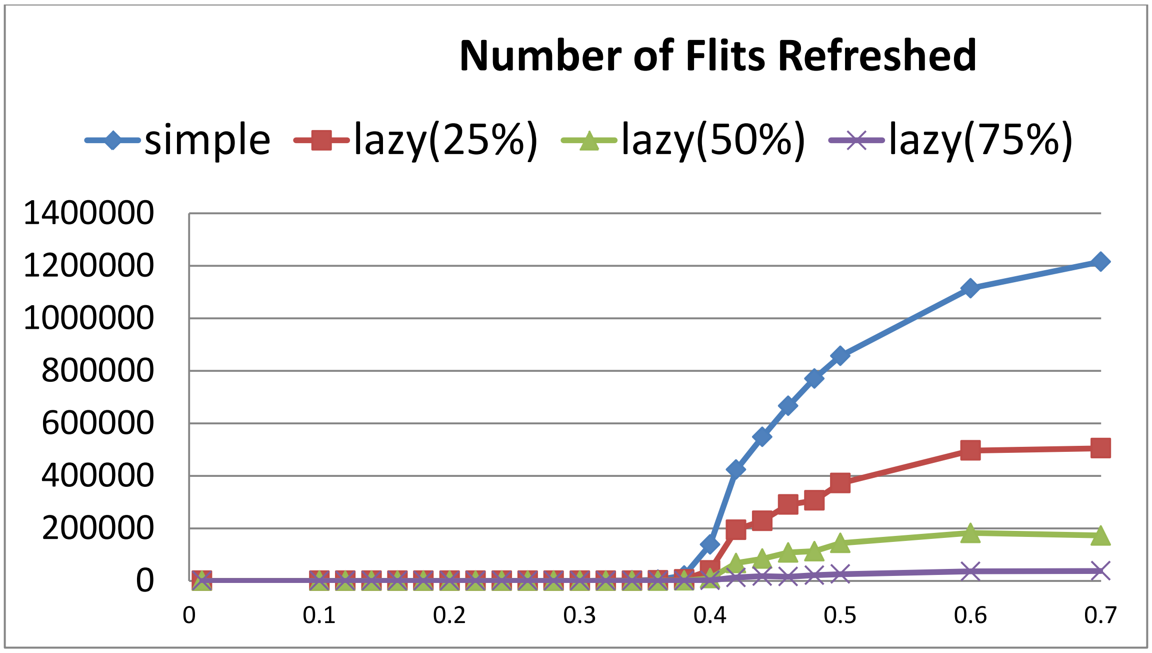


Figure 23. Number of flits refreshed in different refreshing schemes

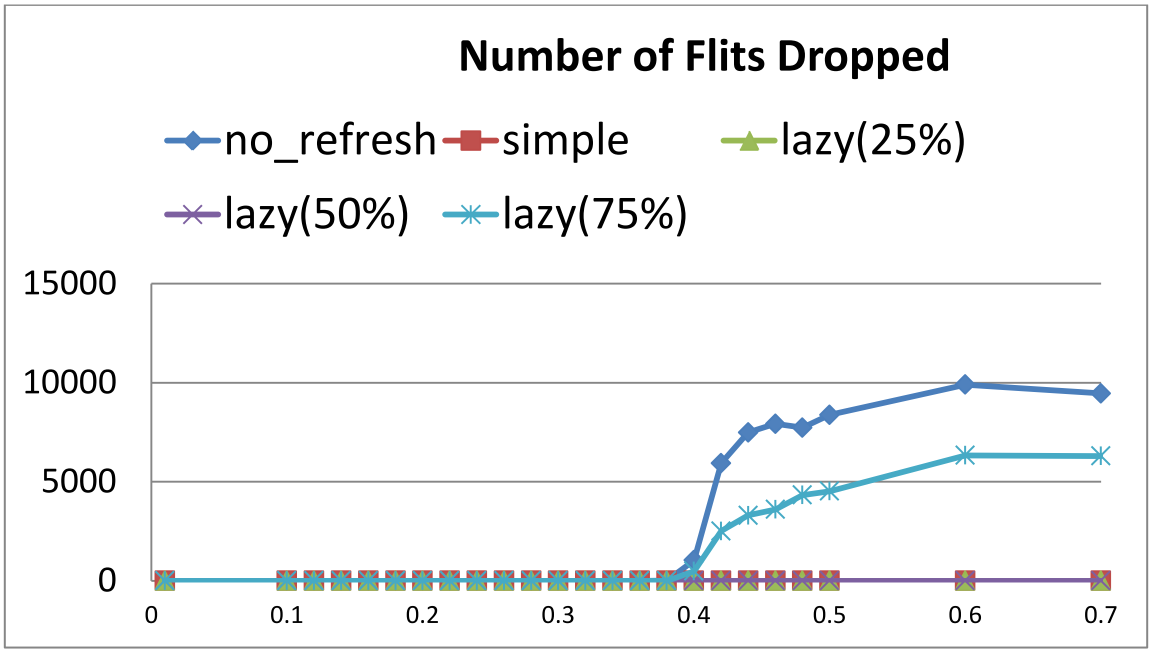


Figure 24. Number of flits dropped in different refreshing schemes

6.3 Area Analysis

We use Orion 2.0 [10] for modeling the area of our STT-MRAM based design. Orion 2.0 provides area estimates for every logic gates. After calculating the area based on the total number of logic components, to account for global white space, it adds 10% to the total area. Table 5 shows the area split by router components for SRAM6, Hybrid SRAM3-STT12 and STT18 router designs.

STT18 design takes about the same area as SRAM6 and outperforms all the designs. Above SRAM 6 flit buffers/vc/port, pure STT-design always has 3x more buffer space advantage and will always perform better than SRAM design before reaching diminishing returns.

In designs with less than 6 flit buffers/vc/port, the fixed logic overhead prevents pure STT-designs to have 3 x buffer space advantages.

Table 5. Estimated area for SRAM6, SRAM3-STT12 and STT18 router designs

	SRAM6(μm^2)	S3-STT12(μm^2)	STT18(μm^2)
Buffer	88241.2	85803.8	87916.9
Crossbar	192087	192087	192087
VCAllocator	8680.52	8680.52	8680.52
SWAllocator	868.05	868.05	868.05
Total	289877	287440	289553

7. CONCLUSIONS

In this thesis, we proposed a pure STT-MRAM based NoC router architecture, and compared it’s performance with pure SRAM and hybrid STT-MRAM based designs. The superior density of STT-MRAM when compared to SRAM and it’s near zero leakage power makes it a suitable alternative to SRAM. Our proposed design solves the challenges involved in incorporating STT-MRAM to NoC routers, like asymmetric write latency and achieves higher throughput compared to conventional SRAM based designs. On an average, we observed throughput improvements of 16% on synthetic workloads and benchmarks. We also looked at the problem of packet dropping and evaluated the effectiveness of Age based SA and refreshing schemes.

REFERENCES

[1] 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.

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

[3] 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.

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

[5] 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.

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

[7] 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.

[8] 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.

[9] 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,” The Pennsylvania State University CSE Dept., Tech. Rep. CSE-11-010, June 2011.

[10] 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.

[11] 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.

[12] B. C. Lee, E. Ipek, O. Mutlu, and D. Burger, “Architecting Phase Change Memory as a Scalable DRAM Alternative,” in *Proceedings of ISCA*, 2009.

[13] P. S. Magnusson, M. Christensson, J. Eskilson, D. Forsgren, G. Hallberg, J. Hogberg, F. Larsson, A. Moestedt, B. Werner, and B. Werner, “Simics: A Full System Simulation Platform,” *Computer*, vol. 35, no. 2, pp. 50–58, 2002.

[14] G. Michelogiannakis, J. D. Balfour, and W. J. Dally, “Elastic-Buffer Flow Control for On-Chip Networks,” in *Proceedings of HPCA*, 2009.

[15] 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.

[16] 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.

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

[18] M. K. Qureshi, M. M. Franceschini, and L. A. Lastras-montaño, “Improving Read Performance of Phase Change Memories via Write Cancellation and Write Pausing,” in *Proceedings of HPCA*, 2010.

[19] M. K. Qureshi, V. Srinivasan, and J. A. Rivers, “Scalable High Performance Main Memory System Using Phase-Change Memory Technology,” in *Proceedings of ISCA*, 2009.

[20] N. D. Rizzo, M. DeHerrera, J. Janesky, B. Engel, J. Slaughter, and S. Tehrani, “Thermally Activated Magnetization Reversal in Submicron Magnetic Tunnel Junctions for Magnetoresistive Random Access Memory,” *Applied Physics Letters*, vol. 80, p. 2335–2337, 2002.

[21] S.C.Woo, M.Ohara, E.Torrie, J.P.Singh, and A.Gupta, “The SPLASH-2 Programs: Characterization and Methodological Considerations,” in *Proceedings of ISCA*, 1995.

[22] 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.

[23] 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.

[24] 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.

[25] Z. Sun, X. Bi, H. H. Li, W.-F. Wong, Z.-L. Ong, X. Zhu, and W. Wu, “Multi Retention Level STT-RAMCache Designs with a Dynamic Refresh Scheme,” in *Proceedings of MICRO*, 2011.

[26] Y. Tamir and G. L. Frazier, "High-Performance Multi-Queue Buffers for VLSI Communications Switches," in *Proceedings of ISCA*, 1988.

[27] 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.

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

[29] H. Zhao, A. Lyle, Y. Zhang, P. K. Amiri, G. Rowlands, Z. Zeng, J. Katine, H. Jiang, K. Galatsis, K. L. Wang, I. N. Krivorotov, and J. P. Wang, "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.

[30] P. Zhou, Y. Du, Y. Zhang, and J. Yang, "Fine-Grained QoS Scheduling for PCM-based Main Memory Systems," in *Proceedings of IPDPS*, 2010.

[31] P. Zhou, B. Zhao, J. Yang, and Y. Zhang, "A Durable and Energy Efficient Main Memory Using Phase Change Memory Technology," in *Proceedings of ISCA*, 2009.

[32] William Dally and Brian Towles. 2003. *Principles and Practices of Interconnection Networks*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.