# Exploit Common Source-Line to Construct Energy Efficient Domain Wall Memory based Caches

Xianwei Zhang*, Lei Zhao* and Youtao Zhang* and Jun Yang†

*Computer Science Department, University of Pittsburgh, Pittsburgh, PA, USA
{xianeizhang, leizhao, zhangyt}@cs.pitt.edu
†Electrical and Computer Engineering Department, University of Pittsburgh, Pittsburgh, PA, USA
juy9@pitt.edu

***Abstract*—Domain wall memory (DWM) is an emerging memory technology that utilizes magnetic domains along a nanowire to achieve high density, short latency and low power. Recent studies showed that it is promising to replace SRAM and STT-MRAM to construct DWM based on-chip caches. However, accessing DWM requires frequent shift operations, which leads to large energy consumption for DWM caches.**

**In this paper, we propose DWM-SSL, an architectural innovation to achieve energy efficiency for multiple-head based DWM caches. DWM-SSL adopts common source line design to re-organize DWM cell arrays such that accessing an N-bit cache line from M-head DWM based cache activates N/M tracks instead of N tracks in the baseline. Our experimental results show that, on average, DWM-SSL reduces around 5.1x track shifts and up to 63% cache energy consumption for a 4-head DWM cache design.**

***Index Terms*—Domain wall memory, last level cache**

## I. Introduction

With the wide adoption of chip-multiprocessors in modern computer systems, there is an increasing demand for large on-chip caches in order to achieve scalable performance. Traditional memory technologies such as SRAM and DRAM face density, scalability, and leakage issues and thus become less appealing for constructing large on-chip caches. Recent works studied emerging memory technologies such as Phase Change Memory (PCM), Spin-Transfer Torque Magnetic RAM (STT-RAM), Resistive RAM (RRAM) and Domain Wall Memory (DWM) [13]. DWM is one of the most promising candidates due to its high density [16], [19], low latency, and low power consumption [19], [18].

DWM improves memory density by storing multiple bits in the magnetic domains along a nanowire, which effectively reduces per bit die area. A nanowire is referred to as a <u>track</u> in the paper. While a track may be equipped with multiple read/write heads, the number of heads is often much smaller than the number of data bits stored in the track. As a result, a DWM access often starts with shifting the desired domain below one read/write head, which introduces both latency and energy overheads. Recent studies on DWM-based last level cache designs focus mainly on designing different head management schemes to reduce the impact of shift operation on cache access latency [18], [20], [16], [19]. As an example, the preshift head management scheme proactively places the head to a domain location that is likely to be accessed [19]. While it reduces the number of shift operations on critical path, such a scheme may introduce extra shift operations in total, resulting in more energy consumption. Recent studies [16] have shown that the shift operations consume more than 50% cache energy for DWM-based last level caches.

The large shift energy consumption comes from the large number of tracks to operate in each DWM access. In particular, accessing a N-bit cache line needs to activate and operate N DWM tracks. This is because, even though a DWM track may have multiple access heads, these heads share both the source line and the bitline such that only one head per track can be active at a time. To address this issue, we propose DWM-SSL, an energy efficient design using common source line array organization. DWM-SSL effectively reduces the number of tracks involved in one access from $N$ to $N/M$. Our contributions are as follows.

- We propose DWM-SSL, a novel track organization for DWM-based cache designs. DWM-SSL places the multiple heads of the same track on different bitlines while the multiple heads of the same index from neighboring tracks share the same source line. This effectively enables the concurrent access of the multiple bits from the same track.
- We propose to exploit DWM-SSL to reduce energy consumption of DWM-based cache designs. DWM-SSL assigns bits from one cache line to the same track such that it effectively reduces the number of tracks to be activated for each cache access.
- We evaluate the reliability of DWM-SSL design and compare the energy consumption to the state-of-the-art DWM-based cache designs. Our results show that, on average, DWM-SSL reduces around 5.1x total track shifts and 63% cache energy consumption for a 4-head DWM cache design.

The rest of this paper is organized as follows: Section II introduces the DWM background. Section III motivates the design and elaborates the details. Section IV and V present the experimental methodology and analyze the results. Related

[1]This work was funded in part by the National Science Foundation under grants CCF-1535755, CCF-1422331 and CNS-1012070.

work is covered in Section VI. We conclude the paper in Section VII.

## II. Background

Domain wall memory (DWM) is an emerging non-volatile memory technology that exploits spintronic magnetoresistive sensing materials and devices. As illustrated in Figure 1, one DWM track (i.e., cell) consists of a ferromagnetic nanowire (NW), one or multiple read/write heads, and two shift ports. NW holds multiple domains separated by ultra narrow domain walls. Each domain represents one binary bit using its magnetic polarity, and several domains share one access port for read and write operations. To enable accesses, the cell is configured with multiple transistors, which dominate the whole area [19].

**DWM Accesses.** To access a magnetic domain, DWM usually needs to inject current pulses ($I_{shift}$) to shift the target domain below a read or write head and then apply an appropriate current ($I_R$ or $I_W$) onto the head to access the domain [13].

The MTJ structure [6], [19], shown in Figure 1, is used to read data from the DWM device. In the MTJ, a ferromagnetic layer with fixed direction is attached to the track such that the resistance of the two layers — the fixed layer and the free layer (i.e., the DMW domain below the fixed layer) — can represent the bit stored in the DWM domain. The domain stores bit 0 if it has the same magnetic orientation as that of the fixed layer and thus the low resistance; or bit 1 if it has anti-parallel orientation and thus high resistance.

There are two DWM write operation implementations [20]. The first one was to perform spin transfer torque, similar to that in STT-MRAM, to alter the data bit saved in one magnetic domain. The other one is to perform a shift-based write, which is to place two fixed domains next to the nanowire. The sandwiched structure allows injecting the current to shift the fixed domains to the free domain in the nanowire, as shown in Figure 1. In this paper, we adopt the latter design as it is not only faster but also highly energy efficient [20].

To minimize the number of shift operations required in read/write accesses, several head selection and management policies [18], [19] have been devised. Among them, *Static* policy statically accesses each bit with a dedicated head, and *Dynamic* always utilizes the runtime closest head to perform bit access. After finishing the access, whereas *Lazy* management policy retains the head at the last access location, *Eager* adversely always restores the head back to its original position which incurs additional shift operations. Putting together the head selection and management policies, we can get four different head policies, *Static-Eager*, *Static-Lazy*, *Dynamic-Eager* and *Dynamic Lazy*. Moreover, swapping mechanism [16] has also been used to further reduce shifts.

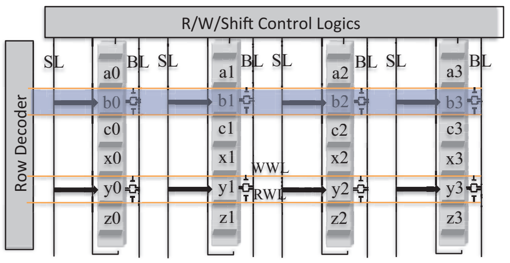
**DWM-based caches.** Recent studies have proposed to architect DWM as the data array of on-chip caches [18], [15]. For a better tradeoff among performance, energy, and area overhead, the tag array is constructed using either SRAM or STT-RAM or 1-bit DWM [18], [20], [19]. Assume we build a 32-way set associative, 64B cache line size on-chip cache using 32-bit long, 4-head DWM tracks. Existing designs always store the 512 bits of each cache line in 512 separate tracks. Accessing a cache line needs to drive all 512 tracks simultaneously. For the 32 ways within one cache set, their corresponding bits, e.g, the first bit of 32 different ways, may be stored in the same track [18], or in four different tracks (with 8 ways per track) [15]. For the latter, the bits for the 8 ways on one track are stored consecutively such that they are to be accessed using one head.

## III. The Design

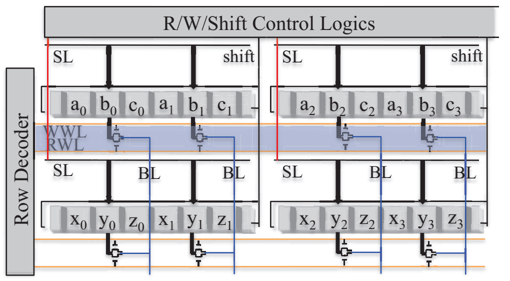
In this section, we first motivate the design by analyzing the array organization of DWM-based caches. We then propose to share bitlines across the tracks. We then analyze its impact on array organization.

### A. The motivation

DWM is a type of bipolar nonvolatile memory and thus needs two bitlines, i.e., bitline (BL) and sourceline (SL), for cell accesses [4], [14]. This is similar to STT-MRAM and memristor. The wordline (WL) in DWM is split into one read wordline (RWL) and one write wordline (WWL) [10], [19], [14]. To read a cell, the read word-line (RWL) transistor is turned on and BL is driven high, the sensing current flowing over the head varies depending on the derived resistance [14]. During write operation, the write word-line (WWL) transistor is on, different voltage conditions (1: SL-low/BL-high, 0: SL-high/BL-low) are created between BL and SL to finish the write by shifting the appropriate fixed domain sided to NW into the target domain.



(a) Conventional bit-interleaved DWM track cluster organization


R/W/Shift Control Logics
Row Decoder
SL
shift
BL
WWL
RWL
$a_0$ $b_0$ $c_0$ $a_1$ $b_1$ $c_1$
$a_2$ $b_2$ $c_2$ $a_3$ $b_3$ $c_3$
$x_0$ $y_0$ $z_0$ $x_1$ $y_1$ $z_1$
$x_2$ $y_2$ $z_2$ $x_3$ $y_3$ $z_3$

(b) Proposed decoupled layout

Fig. 2: DWM layout comparison. 'm' are cache lines, and '$m_i$' are the bits of the corresponding cache line.

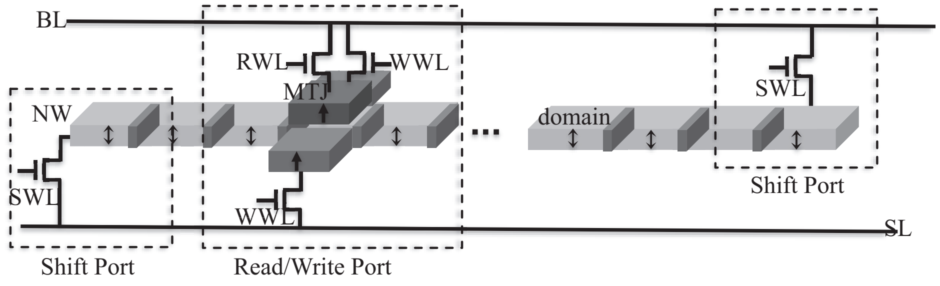
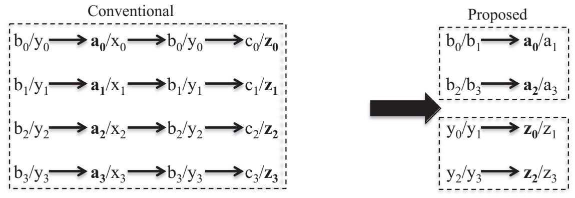


Fig. 1: The structure of a DWM cell.

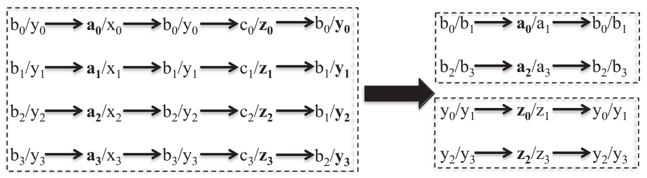
In the existing multi-head DWM designs, the multiple heads of one track share the same BL and SL, which leads to only one head per track being active at a time. On the other hand, a shift operation shifts all domains in a track simultaneously. Figure 2 compares possible cell array reorganizations. For illustration purpose, we assume one cache line has 4 bits (e.g., cache line 'a' is composed of four bits 'a0' - 'a3'), and each DWM track stores 6 bits and has 2 head ports. Given that the distance between 'b' and 'y' is the same as that between two heads, moving 'b' below one head automatically moves 'y' below another head.

Let us assume we need to first access cache line 'a' and then 'z', and the heads' default positions are right above 'b' and 'y'. As shown in Figure 3, the shift steps involved in the access are denoted as 'm'/'n', where 'm' and 'n' are the cache lines shown in Figure 2, 'm' is the cache line under top head and 'n' under the bottom head. For the lazy head management [19] that the head is left at its last position, the access sequence is as illustrated in left part of Figure 3(a). Clearly, we need to shift 4 heads (1+2) positions, i.e., 12 shifts in total. For the eager head management policy [19], [18], i.e., the head is moved back to its predetermined position after each access, the access sequence is as shown in left part of Figure 3(b), which needs to move 4 heads (1+1+1+1) positions, i.e. 16 total shifts. Note that whereas the last shift operation of 'c'/'**z**'$\rightarrow$'b'/'y' does not count towards access latency, it still contributes to energy consumption.

Inspired by [23] and [9], we present an alternative cell array organization, as shown in Figure 2(b). In such design, multiple heads of a single track are vertically connected to different bitlines while sharing the same source line. The new layout eliminates the conventional contention on the shared bitline and thus enables concurrent accesses via different heads. In addition, a number of neighboring tracks are horizontally configured to share the same source line for area occupancy saving. Given this organization, we store two bits from the same cache line on the same track, e.g., bits '$a_0$' and '$a_1$' reside on the upper-left track in Figure 2. To access a cache line, we now need to activate only two tracks, i.e., operations to cache lines 'a/b/c' and 'x/y/z' are decoupled. If we follow the previously discussed head default positions, then accessing 'a' and 'z' would need to move heads for two tracks for (1+1) and (1+1+1+1) positions respectively, as illustrated in right part of


Conventional
$b_0/y_0 \rightarrow \mathbf{a_0}/x_0 \rightarrow b_0/y_0 \rightarrow c_0/\mathbf{z_0}$
$b_1/y_1 \rightarrow \mathbf{a_1}/x_1 \rightarrow b_1/y_1 \rightarrow c_1/\mathbf{z_1}$
$b_2/y_2 \rightarrow \mathbf{a_2}/x_2 \rightarrow b_2/y_2 \rightarrow c_2/\mathbf{z_2}$
$b_3/y_3 \rightarrow \mathbf{a_3}/x_3 \rightarrow b_3/y_3 \rightarrow c_3/\mathbf{z_3}$
Proposed
$b_0/b_1 \rightarrow \mathbf{a_0}/a_1$
$b_2/b_3 \rightarrow \mathbf{a_2}/a_3$
$y_0/y_1 \rightarrow \mathbf{z_0}/z_1$
$y_2/y_3 \rightarrow \mathbf{z_2}/z_3$

(a) Lazy head management


$b_0/y_0 \rightarrow \mathbf{a_0}/x_0 \rightarrow b_0/y_0 \rightarrow c_0/\mathbf{z_0} \rightarrow b_0/\mathbf{y_0}$
$b_1/y_1 \rightarrow \mathbf{a_1}/x_1 \rightarrow b_1/y_1 \rightarrow c_1/\mathbf{z_1} \rightarrow b_1/\mathbf{y_1}$
$b_2/y_2 \rightarrow \mathbf{a_2}/x_2 \rightarrow b_2/y_2 \rightarrow c_2/\mathbf{z_2} \rightarrow b_1/\mathbf{y_2}$
$b_3/y_3 \rightarrow \mathbf{a_3}/x_3 \rightarrow b_3/y_3 \rightarrow c_3/\mathbf{z_3} \rightarrow b_2/\mathbf{y_3}$
$b_0/b_1 \rightarrow \mathbf{a_0}/a_1 \rightarrow b_0/b_1$
$b_2/b_3 \rightarrow \mathbf{a_2}/a_3 \rightarrow b_2/b_3$
$y_0/y_1 \rightarrow \mathbf{z_0}/z_1 \rightarrow y_0/y_1$
$y_2/y_3 \rightarrow \mathbf{z_2}/z_3 \rightarrow y_2/y_3$

(b) Eager head management

Fig. 3: Shift sequence comparison of conventional (left to arrow in figure) and proposed (right to arrow in figure) track layouts to finish the accesses of 'a' and 'z'.

Figure 3. That is, we only need 4 and 8 shifts, translating into a 75% and 50% shift operation reduction, respectively.

### B. Common Source Line (SL) Layout

To realize the common source line design as illustrated in Figure 2(b), we need to carefully layout the cell and its word line and bit/source lines. Figure 4 presents the 3D view and row/column views of the detailed layout, following design rules in [3] and existing layouts in [19], [11].

From the figure, we can see that multiple tracks are organized as small groups, and the tracks within a group share the upright bitlines. To service an access, a row of tracks from different groups are concurrently activated to transfer data via bitlines. Particularly, to drive the track operations, extra spines are equipped to supply the voltage required by source lines.

Each DWM track consists of 32 bits [1] and four heads which are evenly distributed along the track. Three kinds of transistors (read/write/shift) are involved, all of which are based on 32nm technology node. The bit-cell layout, as shown

[1]Total bits on the track are 40 with 32 usable bits and 8 overhead bits [19], [11]. Without specific comment, the "32-bit track" in the later part represents such a 40-bit track.

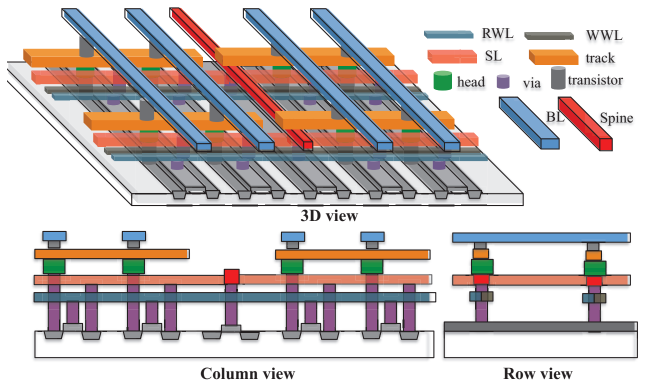
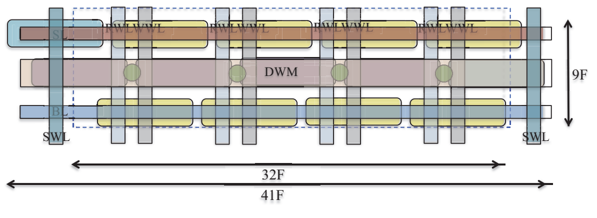
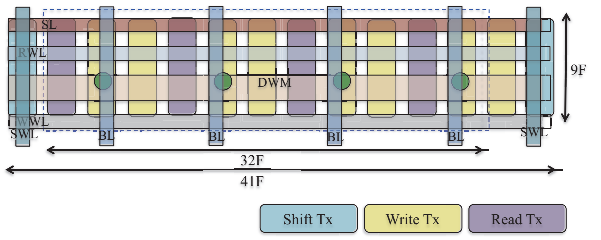


Fig. 4: 3D view of proposed layout

in Figure 5, demonstrates that no extra area is required to function the track.



(a) Existing bit-cell design



(b) Proposed cell design

Fig. 5: DWM bit-cell layout for 32-bit track equipped with 4 heads.

To exploit the new cell array organization, we place $M$ bits (where $M$ is the number of heads of a track) of a cache line on one track, and distribute a $N$-bit cache line in $N/M$ tracks, rather than $N$ tracks in the existing designs.

### C. Impact on Accesses

To read cell values, DWM-SSL applies high voltage to BL and ground SL, which is the same as existing designs. However, the write accesses need to be adjusted accordingly to enable simultaneous accesses to the multiple heads that share the same source line. Given that writing '0' and '1' needs different current directions, we apply voltage 0 to SL, and $+V_{write}$ and $-V_{write}$ to different bitlines, depending if bits '0' or '1' are to be written to individual cells.

The voltage of the common source line, while ideally should be grounded, depends on the new and old values in the track. The cells under heads may be (i) changed from low to high resistance; (ii) changed from high to low resistance; (iii) staying in low resistance state; (iv) staying in high resistance state. The runtime value change pattern thus determines the current distribution in these cells. This is similar to the common source-line design for STT-MRAM [23].

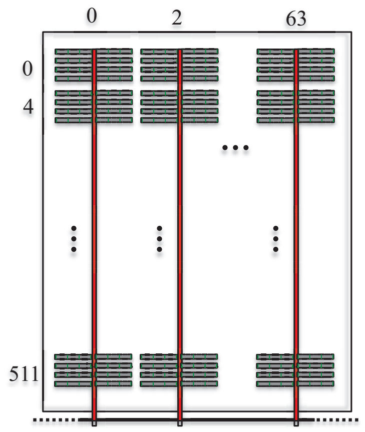


Fig. 6: Array example of the proposed design.

To minimize voltage drop, we connect the 4 heads of one track using one common source line, they are then connected to a spine line, that is laid out vertically along bitline. We illustrate the array organization with spine design in Figure 6, whose 3D layout is shown in Figure 4. For an array consists of 32 DWM cells in one row, we integrate it in the circuit layout, as shown in Figure 6. Following the similar analysis in [23], we find that the voltage drop is negligible when let 4 heads on one DWM track share one common source line.

## IV. Experimental Methodology

In this section, we first present the experimental infrastructure used to evaluate the cache design, and then analyze the obtained cache parameters including area, access latency and energy.

### A. Experimental Setup

To evaluate the effectiveness of our proposed design, we compared it against existing ones using an in-house memory system simulator. We modeled a quad-core system with the parameters shown in Table I. L2 cache is constructed with 32-bit DWM tracks, which is accessed via four heads. The technology node is 32nm (F) [19], [22], [18]. And, as [18], [19], the cache latency and energy parameters were obtained based on SPICE simulation and the modified CACTI tool [1]. The domain wall shifting energy was calculated from micro-magnetic simulations[18], [4]. The architectural simulations are performed over a wide range of SPEC CPU2006 [2] by running 500 million instructions after fast forwarding the first 5 billion instructions and use the following 200 million to warmup the cache.

### B. Cache Characteristics

Table II lists the cache parameters of both the baseline and our proposed design. From the table, first we can see

TABLE I: System Configuration

| | |
|---|---|
| Processor | four 2Ghz OoO cores; four-issue; 128 ROB size |
| L1 Cache | SRAM, private, write-back<br>16KB I/D, 2-way, 2 cycles<br>64B cache line |
| LLC Cache | DWM (32-bit track, 4 heads), shared, write-back<br>2MB, 4 banks, 32-way<br>64B cache line |
| Main Memory | 4GB, DDR3, 200-cycle latency |

TABLE II: Design Parameters for DWM Cache

| | Cell Size | Area | Latency(ns) | | | Energy(nJ) | | |
|---|---|---|---|---|---|---|---|---|
| | ($F^2$) | ($mm^2$) | Rd | Wr | Sh | Rd | Wr | Sh |
| Baseline | 4 | 6.65 | 3.55 | 6.2 | 1.5 | 0.67 | 0.84 | 0.93 |
| Design | 4 | 6.47 | 3.49 | 6.14 | 1.5 | 0.65 | 0.8 | 0.93 |

that cache area has a slight reduction. The explanation is that less tracks are involved in the a cache line access, and hence the decode bits of shift control logic [18] are reduced, which further indicates a reduced decoder area. The same decrease trend can be observed for energy and latency values.

## V. RESULTS

In this section, we first evaluate the performance and energy of DWM-SSL design. And then, we do the sensitivity studies of performance and energy by changing the capacity and associativity of last level cache.

We compare the following "X-Y" schemes. Here, "X" may be one of two cell array layouts.

— Base. The baseline configuration. Each track only stores 1 bit of a cache line.

— SSL. Common source line design. 4 bits of the same cache line are stored in a track.

And "Y" maybe one of the following three recently proposed head management policies [16], [18], [19].

— Lazy. Lazy head management policy. After every access, the head stays on top of the last cell access.

— Eager. Eager head management policy. After every access, the head shifts back to the original position.

— Swap. Enable swapping with Eager head management policy.

### A. Performance Comparison

In the common-source line design, each track can store less number of sets, so every set is separated in different tracks. When accessing a specific cache line of a set, the shift will not affect other sets. This could help keep the locality in each set. Because consecutive addresses are mapped into adjacent sets, various degrees of set level locality exist in different benchmarks. Among the tested benchmarks (Figure 7), SSL-Lazy can achieve up to 6% performance improvement. On average, a 1.6% performance improvement could be achieved.

For both cases (Base- and SSL-) of Eager shift policy, after each access, the track always shifts back to the original position, so SSL's benefit of keeping locality of each set separated in different tracks can no longer be observed, and hence the performance is identical in Base-Eager and SSL-Eager. However, when apply swapping with Eager shift policy, the common-source line design can gain an average of 2.5% performance improvement. This is because sets are separated in different tracks, when two cache lines in the same track need to be swapped, we can simply shift the head to the new hotline and leave it there, so that no actual swapping is needed.

Figure 8 compares the normalized number of shifts of different schemes. Because different sets are stored in separate tracks, DWM-SSL can avoid the affect of the shift operations in one set on the others. For Lazy head management policy, the common source line design can reduce 23% number of shifts on critical path. The 15% reduction of shift operations in SSL-Swap is cause by the elimination of the actual swaps within the same track.

### B. Energy Comparison

To evaluate the efficiency of energy, we break down the cache energy consumption into shift energy, read energy and write energy. Figure 9 shows the energy breakdown in each case. The energy for each shift, read and write operation is summarized in Table II. The difference of the total energy consumption is dominated by the shift energy. As DWM-SSL can store four bits of the same cache line, for a 64B cache line, the total number of tracks that need to be shifted each time is 128 compared to 512 in baseline. From Figure 8, the number of shift operations of SSL-Lazy is 23% of Base-Lazy, and each shift operation needs to shift 4 times more tracks in baseline than SSL, so the total number of shifts is 5.12 times of SSL. Compared to Base, an average of 52%, 61% and 63% energy reduction can be achieved in SSL-Lazy, SSL-Eager and SSL-Swap respectively. Finally, one thing needs to be noted is that our energy saving is not based on a sacrifice of performance. For instance, in Lazy head management policy, *perlbench* can achieve a performance improvement as much as 6%, while at the same time the energy consumption is only 58% of the baseline.

### C. Sensitivity Study

Figure 10 shows the energy consumption when varying the cache capacity and associativity. When associativity is reduced

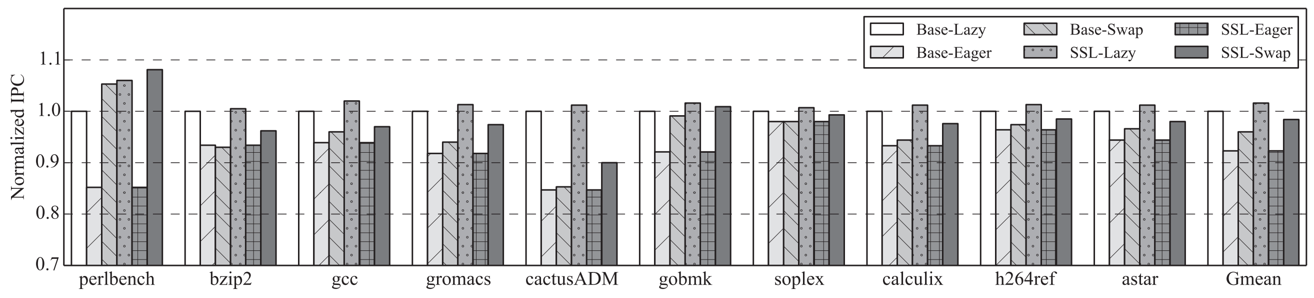


Fig. 7: Performance comparison of different schemes.

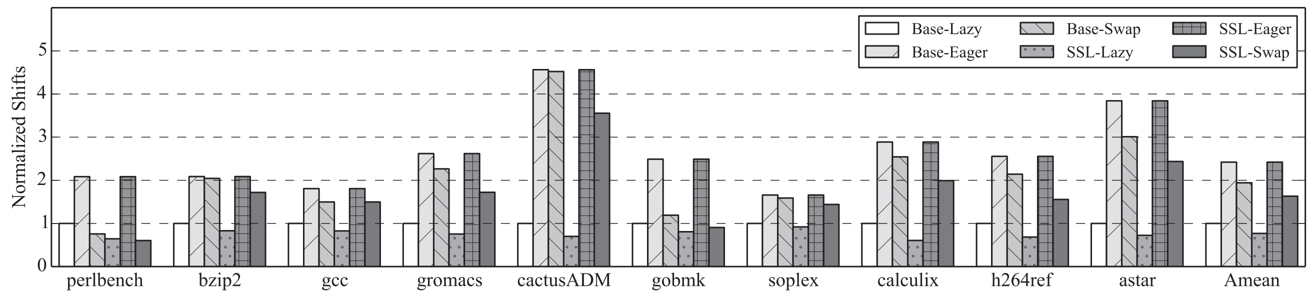


Fig. 8: Shifts on critical path of different schemes.

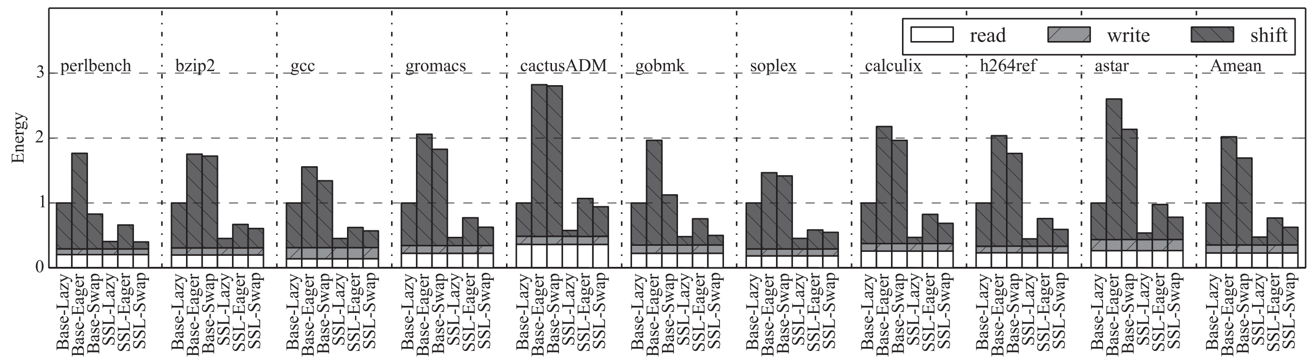


Fig. 9: Energy consumption comparison of different schemes.

to 8-way, all the swaps are within the same track which will induce more additional shifts in the baseline. While in the common source line design, the within-track swaps are actually avoided by leaving the head on the new hotline, so the swapping energy still remains small. This explains why the swap energy in baseline increases. In the 8-way associative configuration, the energy reduction of `SSL-Swap` compared to `Base-Swap` increases to 68%.

IPCs in different capacity and associativity LLCs are compared in Figure 11. When the associativity decreases to 8-way, for the same reason as in the energy analysis, more additional shifts are induced, so that the performance of `Base-Swap` also drops significantly. However, because the common source line design can avoid additional shifts when swap occurs

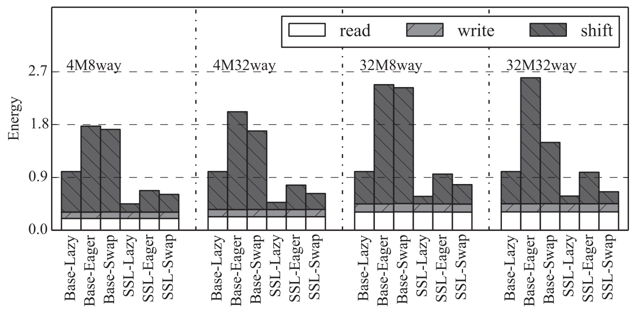


Fig. 10: Energy of caches with different capacity and associativity

within the same track, in the 32M configuration, there is still an IPC increase in `SSL-Swap` over `Base-Swap` when the associativity is decreased to 8-way (5.8%).

When associativity stays the same while increasing the capacity to 32M, we can see from Figure 10 and Figure 11, both the energy and performance of `Eager` head management policy become worse. This is because 32M is large enough for most of our tested benchmarks, fewer evictions exists in each set, which means the hotlines change less frequently. As a result, `Eager` head management policy may suffer more shifts than `Lazy` due to its constant shifts to each line. In general, our design can achieve more improvement when LLC capacity is less than the working set.

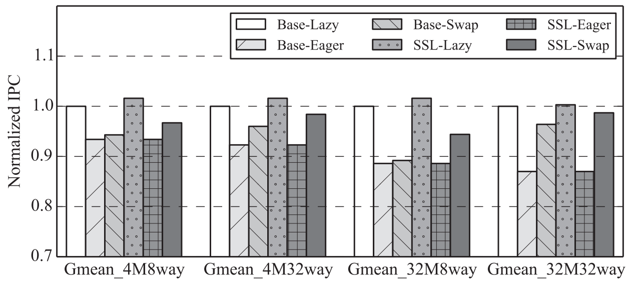


Fig. 11: IPC of caches with different capacity and associativity

## VI. Related Work

Circuit level challenges in domain wall memory such as process variations, joule heating, current generation and shift logic design were addressed in [8], [5], [7], [24], which act as foundations for later work and also ours.

DWM was initially constructed as storage alternative [13], [17], and later demonstrated promising use as on-chip cache. TapeCache [18] was the first attempt to employ DWMs as LLC. Venkatesan *et al.* further proposed DWM-TAPESTRI [20], a new cache design using shift-write DWMs at all level caches. Sun *et al.* [16] optimized TapeCache with new array organization for area efficiency. Motaman *et al.* [11] proposed an energy-efficient and robust DWM cache. In addition to the on-chip cache, recent proposals architect DWM as GPGPU cache [19] or GPGPU register file [10]. DWM has also been explored to accelerate specific applications like BigData[21], encryption [22] and graph computation [12]. Nevertheless, all these designs keep the bitline on a track being shared, and thus only one head can be active, which is just the targeting issue of this paper.

To alleviate the latency and energy penalties of inherent shift operations, the aforementioned works have strived to optimize head organization and managements. Adding multiple heads has been widely used by [20], [18], [16], [10], [15]. Regarding head management, several policies have been studied in [18], [20], [19] to hide the shift penalty. To further moderate the location-dependent shift penalties, people [16], [10], [19] have also seek ways to fully use the domains close to the access ports. Whereas these proposals help to reduce shift operations on critical path, they are highly risky to introduce more shifts in total, and thus the performance improvement is at a sacrifice of energy increase. Differently, our proposed common source-line design aims at the overall energy efficiency without a performance loss.

## VII. Conclusion

In this paper, we proposed a novel cell array organization for DWM based caches. By sharing the same source line for multiple heads on one track, our design significantly reduces the number shift operations in each memory access. Our experimental results show that, on average, we reduce reduces around 5.1x total track shifts and 63% cache energy consumption for a 4-head DWM cache design.

## References

[1] Cacti. http://www.hpl.hp.com/research/cacti/.
[2] SPEC CPU 2006. https://www.spec.org/cpu2006/.
[3] *Computer aids for VLSI design*. Addison-Wesly Publishing Company, 1994.
[4] C. Augustine, A. Raychowdhury, et al. Numerical analysis of domain wall propagation for dense memory arrays. In *IEDM*, 2011.
[5] N. Ben-Romdhane, W. S. Zhao, et al. Design and analysis of racetrack memory based on magnetic domain wall motion in nanowires. In *NANOARCH*, 2014.
[6] S. Fukami, T. Suzuki, et al. Low-current perpendicular domain wall motion cell for scalable high-speed MRAM. In *VLSIT*, 2009.
[7] S. Ghosh. Design methodologies for high density domain wall memory. In *NANOARCH*, 2013.
[8] A. Iyengar and S. Ghosh. Modeling and analysis of domain wall dynamics for robust and low-power embedded memory. In *DAC*, 2014.
[9] C. Kim, K. Kwon, et al. A covalent-bonded cross-coupled current-mode sense amplifier for STT-MRAM with 1T1MTJ common source-line structure array. In *ISSCC*, 2015.
[10] M. Mao, W. Wen, et al. Exploration of GPGPU register file architecture using domain-wall-shift-write based racetrack memory. In *DAC*, 2014.
[11] S. Motaman, A. Iyengar, et al. Synergistic circuit and system design for energy-efficient and robust domain wall caches. In *ISLPED*, 2014.
[12] E. Park, S. Yoo, et al. Accelerating graph computation with racetrack memory and pointer-assisted graph representation. In *DATE*, 2014.
[13] S. Parkin, M. Hayashi, et al. Magnetic domain-wall racetrack memory. In *Science*, 2008.
[14] M. Sharad, R. Venkatesan, et al. Multi-level magnetic RAM using domain wall shift for energy-efficient, high-density caches. In *ISLPED*, 2013.
[15] Z. Sun, X. Bi, et al. Design exploration of racetrack lower-level caches. In *ISLPED*, 2014.
[16] Z. Sun, W. Wu, et al. Cross-layer racetrack memory design for ultra high density and low power consumption. In *DAC*, 2013.
[17] L. Thomas, S.-H. Yang, et al. Racetrack memory: a high-performance, low-cost, non-volatile memory based on magnetic domain walls. In *IEDM*, 2011.
[18] R. Venkatesan, V. Kozhikkottu, et al. Tapecache: A high density, energy efficient cache based on domain wall memory. In *ISLPED*, 2012.
[19] R. Venkatesan, S. G. Ramasubramanium, et al. STAG: Spintronic-take architecture for gpgpu cache hierarchies. In *ISCA*, 2014.
[20] R. Venkatesan, M. Sharad, et al. Dwm-tapestri - an energy efficient all-spin cache using domain wall shift based writes. In *DATE*, 2013.
[21] Y. Wang and H. Yu. An ultralow-power memory-based big-data computing platform by nonvolatile domain-wall nanowire devices. In *ISLPED*, 2013.
[22] Y. Wang, H. Yu, et al. Energy efficient in-memory AES encryption based on nonvolatile domain-wall nanowire. In *DATE*, 2014.
[23] B. Zhao, J. Yang, et al. Common-source-line array: An area efficient memory architecture for bipolar nonvolatile devices. *TODAES*, 18(4), 2013.
[24] W. Zhao, D. Ravelosona, et al. Domain wall shift register-based reconfigurable logic. *IEEE Transactions on Magnetics*, 47(10):2966–2969, 2011.