Area and Performance Co-optimization for Domain Wall Memory in Application-specific Embedded Systems

Shouzhen Gu[1], Edwin H.-M. Sha[1], Qingfeng Zhuge[1], Yiran Chen[2], Jingtong Hu[3]
[1]College of Computer Science, Chongqing University, Chongqing, China
[2]Department of Electrical and Computer Engineering, University of Pittsburgh, Pittsburgh, PA, USA.
[3]School of Electrical and Computer Engineering, Oklahoma State University, OK, USA

ABSTRACT

Domain Wall Memory (DWM), a recently developed spin-based non-volatile memory technology, inherently offers unprecedented benefits in density by storing multiple bits in the domains of a ferromagnetic nanowire, which logically resembles a bit-serial tape. However, this structure also leads to a unique challenge that the bits must be sequentially accessed by performing "shift" operations, resulting in variable and potential higher access latencies. In this paper, we propose a hardware and software co-optimize approach to improve area efficiency and performance for DWM in application-specific embedded systems. For an application-specific embedded system, this technique can obtain a DWM which consists of both micro-cell DWM and macro-cell DWM with minimal area size. Meanwhile, instruction schedule and data allocation with minimal memory access overhead are generated. Experimental results show that the proposed method can minimize the DWM area size while satisfying a system performance constraint.

1. INTRODUCTION

Embedded systems are always in the demand for larger memory within a small size, which has led to the search for denser and more energy-efficient memory technologies that can complement or replace SRAM/DRAM. Emerging memory technologies such as Ferroelectric RAM (FeRAM), Phase Change Memory (PCM), Spin-Transfer Torque Magnetic RAM (STT-MRAM), and Domain Wall Memory (DWM) have been proposed as potential replacements for SRAM as on-chip memory.

Domain Wall Memory (DWM), a recently proposed spin-based memory technology, achieves extremely higher density (around 4X over STT-MRAM) along with ultra-low standby power, and efficient read/write energy with best-case access latencies comparable to SRAM. For these reasons, DWM has captured the attention of academia and semiconductor industry in recent years, and several research efforts have been devoted to understanding the device characteristics [2, 11, 13] and fabricating functional prototypes [5, 9].

Despite possessing a number of favorable features such as very high density, non-volatility, and low leakage power, DWM has unique characteristics that pose significantly different challenges from all other non-volatile memory technologies. There are two different types of DWM cells: micro-cell and macro-cell. In micro-cell DWM, the access speed is fast while the density is low. In macro-cell DWM, the density is high, however the access speed varies. The high density of macro-cell DWM is achieved by sharing access transistors to access multiple magnetic domains where data are sequentially stored in "tapes". Before performing read/write operations, several shift operations are required to align the data with R/W port. Thus, the access speed to macro-cell DWM varies, depending on the number of shift operations required. While the best-case access latencies is close to that in SRAM, the worst-case access latencies can be several times higher.

In order to maximize the benefits of macro-cell DWM, the worst-cases must be avoided. From analysis, we found that the worst cases occurs when two instructions access two data that are on two different ends of the "tape" consecutively. In this case, the maximum number of shift operations are required to access these two data consecutively. By carefully scheduling memory access instructions and placing the data, the worst-cases can be avoided.

Therefore, we propose a hardware and software co-design approach to improve area efficiency and performance for DWM in application-specific embedded systems. For a DWM which consists of macro-cell DWM, we propose an integer non-linear programing (INLP) formulation to achieve the minimum number of shift operations through instruction scheduling and data allocation. Since INLP takes exponential time to finish, we propose a polynomial-time solution, the Instructions Group Schedule (IGS) algorithm, to minimize the number of shift operations. However, the system timing constraint may not be satisfied even though the number of shift operations is minimal. In such cases, it is necessary to adopt a DWM which consists of both micro-cell DWM and macro-cell DWM. The Area and Performance Co-optimization (APCO) algorithm is then proposed to generate a DWM configuration that can satisfy the system performance constraint with minimal area size. The main contributions of this paper include:

- We propose an INLP formulation for instruction scheduling and data allocation to achieve the minimum number of shift operations in DWM that only consists of macro-cells.
- We propose a polynomial-time algorithm, the Instructions Group Schedule (IGS) algorithm, to minimize the number of shift operations.
- We propose the Area and Performance Co-optimization (APCO) algorithm to balance the memory area size and system performance with a hardware and software co-design approach.

According to the experimental results, INLP can reduce

http://dx.doi.org/10.1145/2744769.2744800.

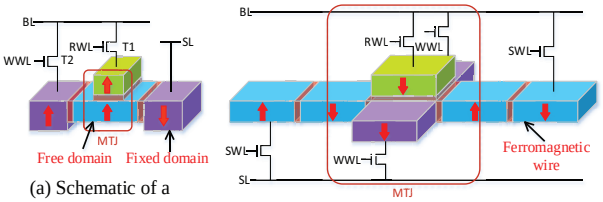
the number of shift operations by 40.26% on average compared with list scheduling. For a DWM which consists of both macro-cells and micro-cells, INLP+APCO can achieve 57.11% improvement in area size on average compared with list+APCO while satisfying performance constraint.

The rest of the paper is organized as follows: in Section 2, background and definitions are introduced. A motivational example is presented in Section 3. The proposed techniques are presented in Section 4. The experiments are presented in Section 5. Related works are discussed in Section 6 and Section 7 concludes the whole paper.

2. BACKGROUND AND DEFINITIONS

When designing DWM based memory for application-specific embedded system, two different cells are considered: micro-cell DWM and marco-cell DWM. A detailed description of these two types of DWMs is presented here.

Micro-cell DWM: As shown in Figure 1(a), the primary components of micro-cell DWM are 2 fixed and 1 free magnetic domains, along with a magnetic tunnel junction (MTJ) sensor. It allows a single bit to be stored in the free domain and read using MTJ. The write operation is carried out by shifting the magnetic orientation of appropriate fixed domain to free domain. The area of DWM is determined by the two transistors. At the 32nm (F) technology node, the area of a micro-cell DWM ($40F^2$) is 3.6X lower than SRAM ($146F^2$) and is comparable to 1T-1R STT-MRAM ($40F^2$) [8].



(a) Schematic of a micro-cell DWM.

(b) Schematic of a macro-cell DWM.

Figure 1: Schematic of micro-cell DWM and macro-cell DWM.

Marco-cell DWM: As shown in Figure 1(b), a macro-cell DWM consists of a ferromagnetic wire, 2 fixed magnetic domains, an MTJ, and 5 access transistors. The read operation is carried out using MTJ and the write operation is carried out by shifting appropriate magnetic orientation to free domain, which is similar to micro-cell DWM. The ferromagnetic wire contains multiple free magnetic domains, such that multiple bits can be stored in the wire. Logically, macro-cell DWM can be viewed as a tape which stores multiple bits of data. Sharing R/W port with multiple bits leads to shift penalty. Before read and write operations, the tape must be shifted to align the bits to the R/W port. Therefore, macro-cell DWM has a shift port, formed by 2 access transistors on the extreme left and right, which is the major difference to micro-cell DWM. At the 32nm (F) technology node, a macro-cell DWM storing 32 bits of data requires $4.5F^2/bit$, achieving $32.4X$ improvement in area compared to SRAM and $9X$ improvement over STT-MRAM.

In this paper, we use Access Instruction Graph (AIG) to model the basic block of an application. The AIG is defined as follows:

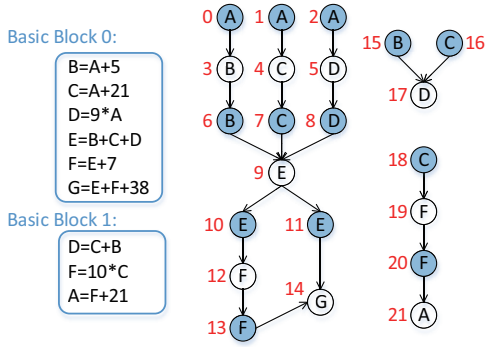
Definition 1. An AIG $G =< V, E, D >$ is a directed graph, where V represents a set of memory access instructions, $E \subseteq V \times V$ is a set of edges to represent the dependencies between memory access instructions, and D is a set of data that is accessed by V.

Formally, the problem addressed by this paper is defined as follows:

Definition 2. Give an AIG G and time constraint T, what is the instruction schedule, data allocation, and DWM macro-cell and micro-cell configuration that can achieve minimum area size for DWM while satisfying a time constraint T.

3. MOTIVATIONAL EXAMPLE

After presenting the definitions, in this section, we will use a motivational example to illustrate the approach proposed in this paper.



(a) A motivational example.

(b) The AIG of motivational example.

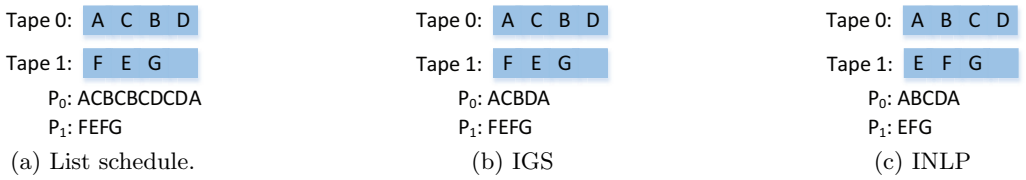
Figure 2: A motivational example and corresponding AIG.

Figure 2(a) shows a motivational example. We assume that basic block 0 and 1 need to access 7 data: A, B, C, D, E, F, and G. The size of each data is 1 bit. Figure 2(b) is the corresponding AIG where the dark circles represent load instructions, and the white circles represent store instructions. For simplicity of illustration, we assign a number to each node as shown beside the node.

In this example, we consider the macro-cell DWM stores 4 bits per tape. Thus, the area of each macro-cell DWM is $4.5 \times 4 = 18F^2$. The area of each micro-cell DWM is $40F^2$. We assume the latency of write operation t_w, read operation t_r, and shift operation t_s of DWM is 10ns, 10ns, and 4ns, respectively. The time constraint T is 250ns.

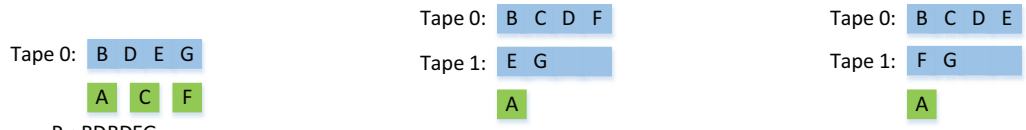
Figure 3 shows the data allocations and schedules generated by three techniques: list schedule, IGS, and INLP, for a DWM which only consists of macro-cells. In Figure 3, the capacity of tape is 4 bits. If we use list schedule, the schedule [0, 1, 2, 15, 16, 18, 3, 4, 5, 17, 19, 6, 7, 8, 20, 9, 21, 10, 11, 12, 14, 13] is generated. Data allocation is shown in Figure 3(a). According to the data allocation and schedule, the track of read/write port of tape 0 and tape 1 is "ACBCBCDCDA" and "FEFG" respectively. Therefore, the number of shift operations is 20. The execution latency is $T1 = t_w * 13 + t_r * 9 + t_s * 20 = 300ns$.

If we use IGS, the data allocation is same as list schedule, as shown in Figure 3(b). The schedule is [0, 1, 2, 15, 18, 4, 7, 16, 3, 6, 5, 17, 8, 19, 20, 21, 9, 10, 11, 12, 13, 14]. The track of read/write port of tape 0 and tape 1 is "ACBDA" and "FEFG" respectively. Therefore, the



(a) List schedule. (b) IGS (c) INLP

Figure 3: The data allocations and schedules generated by list schedule, IGS, and INLP.



(a) List+APCO. (b) IGS+APCO (c) INLP+APCO

Figure 4: The DWM configuration, data allocations, and schedules generated by list+APCO, IGS+APCO, and INLP+APCO.

number of shift operations is 10. The execution latency is $T2 = t_w * 13 + t_r * 9 + t_s * 10 = 260ns$. Compared with list schedule, the number of shift operations is reduced by 50% and the execution latency is reduced by 13.3%. Since instructions that access same data are scheduled adjacently in IGS, shift operations are greatly reduced compared with list schedule.

If we use INLP, the optimal schedule and data allocation with minimum number of shift operations can be generated. The schedule is [0, 1, 2, 16, 3, 6, 15, 18, 4, 7, 5, 17, 8, 9, 10, 11, 12, 13, 19, 20, 14, 21], and the data allocation is shown in Figure 3(c). The track of read/write port of tape 0 and tape 1 are "ACBDA" and "EFG". Therefore, the number of shift operations is 8. The execution latency is $T3 = t_w * 13 + t_r * 9 + t_s * 8 = 252ns$. Compared with list schedule, the number of shift operations is reduced by 60% and the execution latency is reduced by 16.0%.

However, even the performance of optimal solution cannot satisfy the performance constraint $T = 250ns$ with only macro-cell DWM. It is necessary to design a DWM which consists of both macro-cells and micro-cells. Figure 4 shows the DWM configuration, data allocations and schedules generated by list+APCO, IGS+APCO, and INLP+APCO. In Figure 4, the capacity of a micro-cell is 1 bit.

If we use list schedule and APCO together, to satisfy the performance constraint, a DWM with a macro-cell DWM and 3 micro-cell DWMs is generated, as shown in Figure 4(a). The track of read/write port of macro-cell DWM is "BDBDEG". Therefore, the number of shift operations is 5. The area size of DWM is $A1 = 18 * 1 + 40 * 3 = 138F^2$. The execution latency is $T1' = t_w * 13 + t_r * 9 + t_s * 5 = 240ns$.

If we use IGS and APCO together, a DWM with 2 macro-cell DWMs and a micro-cell DWM is generated, as shown in Figure 4(b). The track of read/write port of tape 0 and tape 1 is "BCDF" and "EG". Therefore, the number of shift operations is 4. The area size of DWM is $A1 = 18*2+40*1 = 76F^2$. The execution latency is $T1' = t_w*13+t_r*9+t_s*4 = 236ns$. Compared with list+APCO, the area size is reduced by 44.1%. The number of shift operations and the execution latency are also reduced compared with list+APCO.

If we use INLP and APCO together, a DWM with 2 macro-cell DWMs and a micro-cell DWM is generated, as shown in Figure 4(b). The track of read/write port of tape 0 and tape 1 is "BCDE" and "FG". Therefore, the number of shift operations is 4. The area size of DWM is $A1 = 18 * 2 + 40 * 1 = 76F^2$. The execution latency is $T1' = t_w * 13 + t_r * 9 + t_s * 4 = 236ns$. Compared with list+APCO, the area size is also reduced by 44.1%. The area size, execution latency, and the number of shift operations generated by INLP+APCO are same as IGS+APCO.

4. SOFTWARE AND HARDWARE CO-DESIGN APPROACH

In this section, we present the software and hardware co-design approach. First, we will present the INLP formulations to conduct a schedule and data allocation with minimum number of shift operations. Then, we present Instructions Grouping Schedule (IGS) algorithm. The Area and Performance Co-optimization (APCO) algorithm is presented last.

4.1 INLP Formulations

Before presenting the INLP formulations, we present some notations, in Table 1, which are used in formulations.

Table 1: Notations and Definitions

Notation	Definition
N	The number of nodes in G
$\|E\|$	The number of edges in G
D	A set of data.
K	The number of tapes.
M	The number of cells (bits) of each tape.
$e(u_1, u_2)$	=1, if and only if there is an edge from u_1 to u_2.
$p_{d,k,j}$	=1, if and only if data d is allocated to the j^{th} cell of k^{th} tape.
$a(u, d)$	=1, if and only if data d is accessed by node u.
$var(u)$	The data accessed by node u.
$Sch_{u,k,s}$	=1, if and only if node u accesses tape k at step s , where $0 \leq s < \|N\|$.
$S\_ta(u_1, u_2)$	=1, if and only if nodes u_1 and u_2 access the same tape.

Instruction Constraint. Each memory access instruction must be executed exactly once.

$$\sum_{k=1}^{K}\sum_{s=1}^{N} Sch_{u,k,s} = 1, 1 \leq u \leq N. \quad (1)$$

At each step, there is at most one memory access instruction that accesses tape k.

$$\sum_{u=1}^{N} Sch_{u,k,s} \leq 1, 1 \leq s \leq N, 1 \leq k \leq K. \quad (2)$$

At each step $s(s > 1)$, if there is an instruction executed, then there must be an instruction executed at step $s - 1$.

$$\sum_{u=1}^{N} Sch_{u,k,s} \leq \sum_{u=1}^{N} Sch_{u,k,s-1}, 1 < s \leq N, 1 \leq k \leq K. \quad (3)$$

Dependency Constraint. Let $Ta(u)$ be the tape that instruction u needs to access.

$$Ta(u) = \sum_{k=1}^{K} \sum_{s=1}^{N} Sch_{u,k,s} \times k, 1 \leq u \leq N. \quad (4)$$

$Ta(u)$ can be $1, 2, \ldots, K$.

Let $St(u)$ be the step of instruction u in the Schedule.

$$St(u) = \sum_{k=1}^{K} \sum_{s=1}^{N} Sch_{u,k,s} \times s, 1 \leq u \leq N. \quad (5)$$

$St(u)$ can be $1, 2, \ldots, N$.

For an instruction u, it cannot be executed until all its previous nodes that access same tape are finished. For two instructions u_1 and u_2, $S\_ta(u_1, u_2) = 1$ if and only if $Ta(u_1) = Ta(u_2)$.

$$S\_ta(u_1, u_2) = \begin{cases} 0, & \text{if } Ta(u_1) - Ta(u_2) = 0; \\ 1, & \text{others.} \end{cases} \quad (6)$$

Equation (6) can be transformed into linear constraints. We can formulate the constraint "an instruction u cannot be executed until all its previous nodes that access same tape are finished" as follows

$$(St(u_1) + 1) \leq St(u_2) + S \times S\_ta(u_1, u_2), \forall e(u_1, u_2) \in E. \quad (7)$$

Data Allocation Constraint. Each data must be allocated into a cell of a tape.

$$\sum_{k=1}^{K} \sum_{j=1}^{M} p_{d,k,j} = 1, \forall d \in D. \quad (8)$$

For each cell of each tape, there is at most one data.

$$\sum_{d=1}^{|D|} p_{d,k,j} \leq 1, 1 \leq k \leq K, 1 \leq j \leq M. \quad (9)$$

Let $Alloc(d)$ be the tape where data d is allocated.

$$Alloc(d) = \sum_{k=1}^{K} \sum_{j=1}^{M} p_{d,k,j} \times k, 1 \leq d \leq |D|. \quad (10)$$

Each instruction u needs access the cell where the data d accessed by u is allocated. If $a_{u,d} = 1$, then $Ta(u) = Alloc(d)$.

$$Ta(u) = Alloc(d), \forall a(u, d) = 1. \quad (11)$$

Let $Da(d)$ be the cell where data d is allocated.

$$Da(d) = \sum_{k=1}^{K} \sum_{j=1}^{M} p_{d,k,j} \times j, 1 \leq d \leq |D|. \quad (12)$$

Distance Constraint. For two instructions u_1 and u_2 when $Ta(u_1) = Ta(u_2)$, $St(u_1) - St(u_2) = 1$ represents instruction u_2 will be executed after instruction u_1. Let $Adj(u_1, u_2)$ be a binary variable. We define $Adj(u_1, u_2)$ as follows.

$$Adj(u_1, u_2) = \begin{cases} 0, & \text{if } St(u_1) - St(u_2) = 1; \\ 1, & \text{others.} \end{cases} \quad (13)$$

Equation (13) can be transformed into linear constraints. Let $con(u_1, u_2)$ be a binary variable. $con(u_1, u_2) = 0$ if and only if instruction u_2 will be executed after instruction u_1. We can infer that $con(u_1, u_2) = 0$ if and only if $S\_ta(u_1, u_2) = 0$ and $Adj(u_1, u_2) = 0$.

$$\begin{aligned} &(S\_ta(u_1, u_2) + Adj(u_1, u_2))/2 \leq con(u_1, u_2) \\ &\leq S\_ta(u_1, u_2) + Adj(u_1, u_2). \end{aligned} \quad (14)$$

The distance $Dis(u_1, u_2)$ of two adjacent nodes u_1 and u_2 which access data d_1 and d_2 can be formulated as follows.

$$\begin{aligned} &Dis(u_1, u_2) = |Da(d_1) - Da(d_2)| \times con(u_1, u_2), \\ &d_1 = var(u_1), d_2 = var(u_2), 1 \leq u_1 \leq N, and 1 \leq u_2 \leq N. \end{aligned} \quad (15)$$

Object Function.

$$Min \sum_{u_1=1}^{N} \sum_{u_2=1}^{N} Dis(u_1, u_2) \quad (16)$$

4.2 Instructions Group Schedule Algorithm

Since INLP takes exponential time to finish, we propose a polynomial-time solution, the Instructions Group Schedule (IGS) algorithm as shown in Algorithm 1. The main idea of IGS is that instructions which access same data are scheduled adjacently. The number of shift operations can be reduced in this way.

Algorithm 1 *Instructions Grouping Schedule (IGS) algorithm*

Require: The AIG $G =< V, E, D >$, the number of maro-cell DWMs (tapes): K, the capacity of a maro-cell DWM: M;
Ensure: Schedule Sch, data allocation A, and the number of shift N_s.
1: $P_k \leftarrow$ Record current position of the port of tape k;
2: $N_s \leftarrow$ Record the number of shift operations;
3: $L \leftarrow$ Find executable instructions according to G;
4: **while** (L) **do**
5: $l \leftarrow$ Find an instruction in L which accesses the data pointed by P_1 or P_2 or ... or P_K;
6: **if** l is nonexistent **then**
7: Find an instruction l' where data d accessed by l' has been allocated;
8: **if** l' is nonexistent **then**
9: Select an instruction l_r in L randomly and allocate the data d accessed by l_r to an available tape;
10: **end if**
11: $N_s = N_s + |P_k - pos(d)|$;
12: Update $P_1, P_2, ..., P_K$;
13: **end if**
14: Schedule instruction;
15: Update L;
16: **end while**
17: **return** Sch, A, and N_s

IGS employs array $P = [P_1, P_2, ..., P_K]$ to record the position of access ports, N_s to record the number of shift operations, and list L to store executable instructions. After initialization, it generates instruction schedule Sch and data allocation A from line 4 to 15. In line 5, it finds an

instruction l in L where l needs to access the data pointed by P_1 or P_2 or ... or P_K. If l is nonexistent, in line 7, it finds an instruction l' in L where the data accessed by l' has been allocated. If l' is nonexistent, it finds an instruction l_r in L randomly and allocates the data accessed by l_r to an available tape in line 9. In line 11, it computes the number of shift operations N_s according to equation $N_s = N_s + |P_k - pos(d)|$ where $pos(d)$ is the position of data d in tape k. Since there are shift operations, in line 12, it updates the position of access ports $P_1, P_2, ..., P_K$. In line 14, it schedules current instruction l or l' or l_r. In line 15, it updates list L according to G. The complexity of IGS algorithm is $O(|V|^2 \times |E| \times K \times M)$ where $|V|$ is the number of instructions, $|E|$ is the number of edges, K is the number of tapes, and M is capacity of a macro-cell DWM.

4.3 Area and Performance Co-optimization Algorithm

Macro-cell DWM has high density and inferior performance due to shift operations. To satisfy a performance constraint T, software optimization may be insufficient. In this subsection, we present a software and hardware co-design approach, the Area and Performance Co-optimization (APCO) algorithm as shown in Algorithm 2, to maintain high density while satisfying performance constraint. The main goal of APCO algorithm is to minimize the area of DWM while satisfying the time constraint T.

Algorithm 2 *Area and Performance Co-optimization (APCO) algorithm*

Require: An AIG $G =< N, E, D >$, time constraint T, configuration of macro-cell DWM: M and $Area_{ma}$, and configuration of micro-cell DWM: $Area_{mi}$;
Ensure: An DWM with minimum area.
1: Estimate the number of macro-cell DWM N_{ma};
2: $N_s \leftarrow$ The number of shift operations;
3: Generate $T' = N_w \times t_w + N_r \times t_r + N_s \times t_s$;
4: **while** ($|T' - T| > \alpha$) **do**
5: Generate $Na_s = N_s/N$ and $D_s = (T' - T)/t_s$;
6: Find a set of data D' according to Na_s and D_s;
7: **if** (T'>T) **then**
8: Allocate D' to micro-cell DWM;
9: **else**
10: Allocate D' to macro-cell DWM;
11: **end if**
12: Update G' according to D';
13: $N_s \leftarrow$ The number of shift operations;
14: Generate T';
15: **end while**
16: **return** An DWM: N_{ma}, N_{mi}, and area

In line 1 of Algorithm 2, it estimates the number of macro-cell DWM N_{ma} according to capacity of a macro-cell DWM and D. In line 2, it generates the number of shift operations N_s if only macro-cell DWM is adopted. Here either IGS and INLP can be employed to obtain N_s. In line 3, it computes the total memory access time T' with following equation: $T' = N_w \times t_w + N_r \times t_r + N_s \times t_s$, where N_w is the number of write operations, N_r is the number of read operations, t_w is the latency of write operation, t_r is the latency of read operation, and t_s is the latency of shift operation. If $|T' - T| > \alpha$ where α is a permissible value, it adjusts N_{ma} and N_{mi}. In line 5, it computes the average number of shift operations per instruction Na_s and the number of shift operations that need to be reduced D_s to satisfy time constraint T. In line 6, it finds a set of data D' in macro-cell DWM such that the number of instructions which access data in D' is no less than D_s/Na_s. If $T' > T$, it allocates data in D' to micro-cell DWM. Otherwise, it allocates data in D' to macro-cell DWM. In line 12, it updates G' according to D'. If data in D are moved from macro-cell to micro-cell, it omits D', the instructions that access data in D', and corresponding edges from G'. Otherwise, it adds D', the instructions that access data in D', and corresponding edges to G'. In line 13, it generates N_s. In line 14, it updates the total memory access time T'.

5. EXPERIMENTS

In this section, we present the experimental setup in subsection 5.1. The experimental results and analysis are presented in subsection 5.2.

5.1 Experimental setup

We developed a custom simulator to conduct the experiments. Two sets of experiments are conducted. In the first set, the targeting architecture is equipped with a DWM which consists of 32-bit macro-cells. The performance comparison of list schedule [15], IGS, and INLP is conducted. In the other set, the targeting architecture is equipped with a DWM which consists of 32-bit macro-cell and micro-cell. The density comparison of list+APCO, IGS+APCO, and INLP+APCO is conducted. Benchmarks from Mediabench [4] are adopted. The area size estimation and latency parameters are obtained from NVsim [3].

For the INLP experiments, Lingo is used as solver tool. Not all benchmarks can be solved optimally. We terminate the ILP solver after several minutes and used the best results that solver had.

5.2 Experimental Results and Analysis

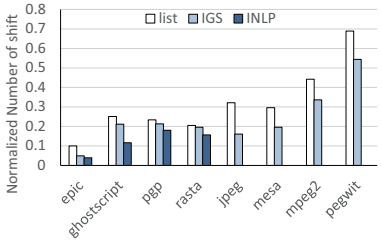


Figure 5: The performance of various techniques.

Figure 5 shows the performance (normalized number of shift) of various techniques on a DWM which consists of macro-cell. Compared with list schedule, IGS can reduce the number of shift operations by 26.09% on average. For "epic" and "jpeg", the numbers of shift operations are reduced by 50% approximately. INLP can reduce the number of shift operations by 40.26% on average. For "epic", the number of shift operations is reduced by 60.96%. According to the experimental results shown in Figure 5, IGS and INLP are efficient in reducing the number of shift operations. For macro-cell DWM, the reduction of shift operations can improve the performance significantly.

Figure 6 shows the area size of DWM (consists of both macro-cell and micro-cell) generated by various techniques under the performance constraint. IGS+APCO can achieve

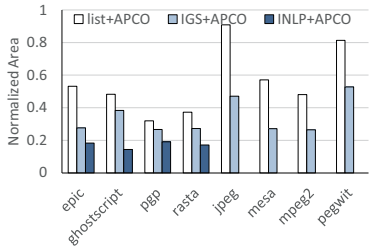


Figure 6: Area size generated by various techniques.

36.60% improvement on average in area size compared to list+APCO. For the best case, IGS+APCO can achieve 52.34% improvement. INLP+APCO can achieve 57.11% improvement on average in area size. Experimental results show that IGS+APCO and INLP+APCO are efficient in generating a DWM with minimal area size while satisfying performance constraint.

6. RELATED WORKS

DWM storage applications based on racetrack memory have been demonstrated, such as the array integration at standard IBM 90nm technology [1] and the content addressable memory (CAM) design and fabrication [7]. Exploiting the tape-like nature, DWM was shown to be well suited for designing on-chip FIFOs [12]. In [9], DWM was also proposed as a potential replacement for secondary storage. These works all show the potential of DWM as a promising memory technology.

DWM has unique characteristics that pose significantly different challenges from all other memory technologies. There are several previous works focusing on the trade-off between density and average access latency in hardware. Techniques such as cache management policies for head selection and update were proposed to address the overhead of shift operations [13, 14] when DWM is used in cache. There are some previous works to improve the performance of DWM with schedule and data allocation techniques. Mao et al. [6] proposed a DWSW-RM aware warp scheduling algorithm to hide the shift delay of DWM. Sun et al. [10] proposed an application-driven data management policy to minimize the racetrack shifting. In this paper, we propose a hardware and software co-design approach to co-optimize area and performance for DWM in application-specific embedded systems. DWM is employed as on-chip scratchpad memory. Therefore, existing techniques cannot be applied in our problem.

7. CONCLUSIONS

In this paper, we propose a hardware and software co-design approach to co-optimize area and performance for DWM in application-specific embedded systems. For a DWM which consists of macro-cell DWM, an INLP formulation and the IGS algorithm are proposed to achieve the minimum number of shift operations. For DWM consists of both micro-cell and macro-cell, the APCO algorithm is proposed to generate a DWM configuration that can satisfy the system performance constraint with minimal area size. Experimental results confirm the effectiveness of the proposed approach.

8. ACKNOWLEDGMENTS

This work is partially supported by NSF CNS-1116171, CNS-1464429, National Science Foundation of China 61472052, 61173014, National High Technology Research and Development Program of China (863 Program) 2013AA013202, Chongqing High-Tech Research Program cstc2012ggC40005.

9. REFERENCES

[1] A. Annunziata, M. Gaidis, L. Thomas, C. Chien, C. Hung, P. Chevalier, E. O'Sullivan, J. Hummel, E. Joseph, Y. Zhu, et al. Racetrack memory cell array with integrated magnetic tunnel junction readout. In *Proc. IEDM*, pages 24–3. IEEE, 2011.

[2] C. Augustine, A. Raychowdhury, B. Behin-Aein, S. Srinivasan, J. Tschanz, V. K. De, and K. Roy. Numerical analysis of domain wall propagation for dense memory arrays. In *Proc. IEDM*, pages 17.6.1–17.6.4, Dec. 2011.

[3] X. Dong, C. Xu, Y. Xie, and N. P. Jouppi. Nvsim: A circuit-level performance, energy, and area model for emerging nonvolatile memory. *IEEE TCAD*, 31(7):994–1007, 2012.

[4] C. Lee, M. Potkonjak, and W. H. Mangione-Smith. Mediabench: a tool for evaluating and synthesizing multimedia and communicatons systems. In *Proceedings of the 30th annual ACM/IEEE international symposium on Microarchitecture*, pages 330–335. IEEE Computer Society, 1997.

[5] E. R. Lewis, D. Petit, L. O'Brien, A. Fernandez-Pacheco, J. Sampaio, A.-V. Jausovec, H. T. Zeng, D. E. Read, and R. P. Cowburn. Fast domain wall motion in magnetic comb structures. *Nature*, 9(12):980–983, Dec. 2010.

[6] M. Mao, W. Wen, Y. Zhang, Y. Chen, and H. H. Li. Exploration of gpgpu register file architecture using domain-wall-shift-write based racetrack memory. In *Proc. DAC*, pages 1–6. ACM, 2014.

[7] R. Nebashi, N. Sakimura, Y. Tsuji, S. Fukami, H. Honjo, S. Saito, S. Miura, N. Ishiwata, K. Kinoshita, T. Hanyu, et al. A content addressable memory using magnetic domain wall motion cells. In *VLSIC*, pages 300–301. IEEE, 2011.

[8] S. P. Park, S. Gupta, N. Mojumder, A. Raghunathan, and K. Roy. Future cache design using stt mrams for improved energy efficiency: Devices, circuits and architecture. In *Proc. DAC*, pages 492–497, Jun. 2012.

[9] S. S. P. Parkin, M. Hayashi, and L. Thomas. Magnetic domain-wall racetrack memory. *Science*, 320(5873):190–194, Apr. 2008.

[10] Z. Sun, X. Bi, W. Wu, S. Yoo, and H. H. Li. Array organization and data management exploration in racetrack memory. *TC*, PP(99), Sep. 2014.

[11] L. Thomas, R. Moriya, C. Rettner, and S. S. P. Parkin. Dynamics of magnetic domain walls under their own inertia. *Science*, 330(6012):1810–1813, Dec. 2010.

[12] R. Venkatesan, V. K. Chippa, C. Augustine, K. Roy, and A. Raghunathan. Energy efficient many-core processor for recognition and mining using spin-based memory. In *Proc. NANOARCH*, pages 122–128. IEEE Computer Society, 2011.

[13] R. Venkatesan, M. Sharad, K. Roy, and A. Raghunathan. Dwm-tapestri - an energy efficient all-spin cache using domain wall shift based writes. In *Proc. DATE*, pages 1825–1830, Apr. 2013.

[14] R. Venkatesany, V. Kozhikkottuy, C. Augustinex, A. Raychowdhuryx, K. Royy, and A. Raghunathan. Tapecache: A high density, energy efficient cache based on domain wall memory. In *Proc. ISLPED*, pages 185–190, Jul. 2012.

[15] T. Yang and A. Gerasoulis. List scheduling with and without communication delays. *Parallel Computing*, 19(12):1321–1344, 1993.