An ultralow-power memory-based big-data computing platform by nonvolatile domain-wall nanowire devices

Wang, Yuhao; Yu, Hao

2013

Wang, Y., & Yu, H. (2013). An ultralow-power memory-based big-data computing platform by nonvolatile domain-wall nanowire devices. IEEE International Symposium on Low Power Electronics and Design (ISLPED) 2013.

<https://hdl.handle.net/10356/100920>

© 2013 IEEE. This is the author created version of a work that has been peer reviewed and accepted for publication by IEEE International Symposium on Low Power Electronics and Design (ISLPED) 2013, IEEE. It incorporates referee's comments but changes resulting from the publishing process, such as copyediting, structural formatting, may not be reflected in this document. The published version is available at:
<http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=6629318&url=http%3A%2F%2Fieeexplore.ieee.org%2Fexpl>

Downloaded on 26 Apr 2023 15:54:09 SGT

An Ultralow-power Memory-based Big-data Computing Platform by Nonvolatile Domain-wall Nanowire Devices

Yuhao Wang and Hao Yu

School of Electrical and Electronic Engineering,
Nanyang Technological University, Singapore 639798
haoyu@ntu.edu.sg

Abstract—As one recently introduced non-volatile memory (NVM) device, domain-wall nanowire (or race-track) has shown potential for main memory storage but also computing capability. In this paper, the domain-wall nanowire is studied for a memory-based computing platform towards ultra-low-power big-data processing. One domain-wall nanowire based logic-in-memory architecture is proposed for big-data processing, where the domain-wall nanowire memory is deployed as main memory for data storage as well as XOR-logic for comparison and addition operations. The domain-wall nanowire based logic-in-memory circuits are evaluated by SPICE-level verifications. Further evaluated by applications of general-purpose SPEC2006 benchmark and also web-searching oriented Phoenix benchmark, the proposed computing platform can exhibit a significant power saving on both main memory and ALU under the similar performance when compared to CMOS based designs.

I. INTRODUCTION

The analysis of big-data at exascale (10^{18} bytes or flops) has introduced the emerging need to reexamine the existing hardware platform that can support memory-oriented computing. A big-data-driven application requires huge bandwidth with maintained low-power density. For example, web-searching application involves crawling, comparing, ranking, and paging of billions of web-pages with extensive memory access [1]. However, the current data-processing platform has well-known memory-wall with limited accessing bandwidth but also large leakage power at advanced CMOS technology nodes. As such, a power-efficient memory-based design is highly desirable for future big-data processing. Towards this end, there are many recent explorations by the newly discovered non-volatile memory (NVM) technologies such as phase-change memory (PCM), spin-transfer torque memory (STT-RAM), and resistive memory (ReRAM) [2], [3], [4], [5], [6], [7], [8].

The primary advantage of NVM is the potential as the universal memory with significantly reduced leakage power. For example, STT-RAM is considered as the second-generation of spin-based memory, which has sub-nanosecond magnetization switching time and sub-pJ switching energy [9], [10], [11]. As the third-generation of spin-based memory, domain-wall nanowire, also known as racetrack memory [12], [13], is a newly introduced NVM device that can have multiple bits densely packed in one single nanowire, where each bit can be accessed by the manipulation of the domain-wall. Compared with STT-RAM, the domain-wall nanowire is able to provide the similar speed and power but with much higher density or throughput [14]. Since domain-wall nanowire has close-to-DRAM density but with close-to-zero standby power, it becomes an ideal candidate for future main memory that can be utilized for big-data processing.

However, there is no in-depth study to explore domain-wall nanowire based computing platform towards big-data processing. For example, no link has been made to perform big-data logic operation based on spin-based device such as domain-wall nanowire. What is more, no domain-wall nanowire device model has been developed in

terms of accuracy and efficiency for circuit designs. In this paper, we show that shift-operation in domain-wall nanowire can introduce unique capability to perform logic operations which other NVM devices do not have. Due to the low operating power and also zero standby power, the logic operations such as XOR-logic for comparison and addition by domain-wall nanowire can be performed in an ultra-low power fashion towards big-data applications such as web-searching. As such, one can design a memory-based computing platform with both memory and logic by domain-wall nanowire devices. In addition, a SPICE behavior model of domain-wall nanowire has been developed for circuit-level verification of both memory and logic designs, which further provide system-level evaluation of area, timing and power. The numerical experiments show that, with the use of applications of general-purpose SPEC2006 benchmark and also web-searching oriented Phoenix benchmark, the proposed memory-based computing platform by domain-wall nanowire can reduce 92% leakage power and 16% dynamic power when compared with main memory implemented by DRAM; and can reduce 31% both dynamic and 65% leakage power under the similar performance when compared with ALU implemented by their CMOS counterpart.

The rest of this paper is organized in the following manner. Section II introduces the overall memory-based computing platform based on domain-wall nanowire. Section III discusses the SPICE model of domain-wall nanowire. Section IV describes the main memory design by domain-wall nanowire. Section V presents the XOR-logic for comparison and addition by domain-wall nanowire. Experimental results are presented in Section VI with conclusion in Section VII.

II. MEMORY-BASED BIG-DATA COMPUTING PLATFORM

Fig. 1: The overview of the big data computing platform by domain wall nanowire devices

The overview of the proposed big-data computing platform is shown in Figure 1. The big-data applications are compiled by MapReduce-based parallel-computing model to generate scheduled tasks. A memory-based computing system is organized with integrated many-core microprocessor and main memory, which are mainly composed of the non-volatile domain-wall nanowire devices. The many-core microprocessors are further classified into clusters. Each cluster shares L2-cache and accesses the main memory by

shared memory bus. Each core works highly independently for allocated tasks such as Map or Reduce functions.

In this paper, the domain-wall nanowire is intensively utilized towards the ultra-low power big-data processing in both memory and logic, simultaneously. The domain-wall nanowire based main memory can significantly reduce both the leakage and operating power of the main memory. What is more, large-volume of memory can be integrated with high density for data-driven applications. As such, one can build a hybrid memory system with CMOS-based cache as well as domain-wall nanowire based main memory, which composition can be optimized by studying the accessing patterns under big-data applications. More importantly, the domain-wall nanowire is also explored for computing purpose in this paper. Specifically, the domain-wall nanowire based XOR-logic for comparison and addition is studied in details based on the following observations. Firstly, at instruction level, the web-searching orientated big-data applications usually involve intensive string operations, namely comparison, where the XOR and Adder logics will be visited more frequently than usual. Moreover, from logic level, the transistors to implement XOR gates in ALU account for more than half of the total number, due to its much higher complexity compared to the NAND, NOR and NOT gates. As such, an optimized design of XOR-logic by new technology such as domain-wall nanowire may provide the largest margin to optimize hardware for big-data processing.

III. DEVICE MODELING AND SIMULATION

Domain-wall nanowire, also known as racetrack memory [12], is a newly introduced non-volatile memory device in which multiple bits of information are stored in single ferromagnetic nanowire. As shown in Figure 2(a), each bit is denoted by the leftward or rightward magnetization direction, and adjacent bits are separated by domain walls. By applying a current through the shift port at the two ends of the nanowire, all the domain walls will move left or right at the same velocity while the domain width of each bit remains unchanged, thus the stored information is preserved. Such a tape-like operation will shift all the bits similarly like a shift register.

In order to access the information stored in the domains, a strongly magnetized ferromagnetic layer is placed at desired position of the ferromagnetic nanowire and is separated by an insulator layer. Such a sandwich-like structure forms a magnetic-tunnel-junction (MTJ), through which the stored information can be accessed. In the following, the write, read and shift operations are modeled respectively.

Fig. 2: (a) Schematic of domain-wall nanowire structure with access port and shift port; (b) magnetization of free-layer in spherical coordinates with defined magnetization angles; and (c) typical R-V curve for MTJ

A. Magnetization reversal

The write access can be modeled as the magnetization reversal of MTJ free layer, i.e. the target domain of the nanowire. Note

that the dynamics of magnetization reversal can be described by the precession of normalized magnetization m , or state variables θ and ϕ in spherical coordinates as shown in figure 2(b). The spin-current induced magnetization dynamics described by θ and ϕ is given by [15]

$$\dot{\theta} = \theta_0 \text{Exp} \left(-\frac{t}{t_0} \right) \cdot \cos(\phi) \quad (1)$$

$$\dot{\phi} = \frac{d\phi}{dt} = k_1 \sqrt{k_2 - (k_3 - k_4 I)^2} \quad (2)$$

where θ_0 is the initial value of θ , slightly tilted from the stable x or $-x$ directions; t_0 is precession time constant; ω is the angular speed of ϕ ; k_1 to k_4 are magnetic parameters with detailed explanation in [15]; and I is the spin-current that causes the magnetization precession.

B. Magnetic-tunnel-junction resistance

A typical R-V curve for MTJ is shown in Figure 2(c) with two regions: giant magnetoresistance (GMR) region and tunneling region. Depending on the alignment of magnetization directions of the fixed layer and free layer, parallel or anti-parallel, the MTJ exhibits two resistance values R_l and R_h . As such, the general MTJ resistance can be calculated by the giant magnetoresistance (GMR) effect

$$R(\theta_u, \theta_b) = R_{l0} + \frac{R_{h0} - R_{l0}}{2} (1 - \cos(\theta_u - \theta_b)) \quad (3)$$

where θ_u and θ_b are the magnetization angles of upper free layer and bottom fixed layer, R_{l0} and R_{h0} are the MTJ resistances when the applied voltage is subtle. When the applied voltage increases, there exists tunneling effect caused voltage-dependent resistance roll-off,

$$\begin{cases} R_l(V) = \frac{R_{l0}}{1 + c_l V^2} \\ R_h(V) = \frac{R_{h0}}{1 + c_h V^2} \end{cases} \quad (4)$$

where c_l and c_h are voltage-dependent coefficients for parallel state and anti-parallel states, respectively.

C. Domain-wall propagation

Like a shift register, the domain-wall nanowire shifts in a digital manner, thus could be digitalized and modeled in the unit of domains, in which a bit is stored. Note that except the bit in the MTJ, the other bits denoted by the magnetization directions are only affected by their adjacent bits. In other words, the magnetization of each bit is controlled by the magnetization in adjacent domains. Inspired by this, we present a magnetization controlled magnetization (MCM) devices based behavioral model for domain-wall nanowires. Unlike the current-controlled and voltage-controlled devices, the control in MCM device needs to be triggered by rising edge of one SHF-signal, which can be formulated as

$$\begin{aligned} \theta &= f(T_{sl}, \theta_r, T_{sr}, \theta_l, \theta_c) \\ &= T_{sl} \theta_r + T_{sr} \theta_l + \bar{T}_{sl} \bar{T}_{sr} \theta_c. \end{aligned} \quad (5)$$

In which T_{sl} and T_{sr} are the shift-left and shift-right commands; θ_r and θ_l are the magnetization angles in right adjacent cell and left adjacent cell respectively; θ_c is the current state before the trigger signal. This describes that the θ -state will change when triggered and will remain state if no shift-signal is issued.

For the bit in MTJ, the applied voltage for spin-based read and write will also determine the θ -state as discussed previously. Therefore we have,

$$\theta = f(T_{sl}, \theta_r, T_{sr}, \theta_l, \theta_0) + g(V_p, V_n, \theta_c) \quad (6)$$

where V_p and V_n are the MTJ positive and negative nodal voltages, and $g(V_p, V_n, \theta_0)$ is the additional term that combines Equation 1 to 4.

In addition, the domain-wall propagation velocity can be mimicked by the SHF-request frequency. The link between the SHF-request frequency and the propagation velocity is the experimentally observed by current-velocity relation [16],

$$v = k(J - J_0), \quad (7)$$

where J is the injected current density and J_0 is the critical current density.

By combining equations (1) to (6) together, with the magnetization angles θ and ϕ as internal state variables other than electrical voltages and currents, one can fully describe the behaviors of the domain-wall nanowire device, where each domain is modeled as the proposed MCM device. As such, the modified nodal analysis (MNA) can be built in the SPICE-like simulator [17], [18] to verify circuit designs by domain-wall nanowire devices.

IV. DOMAIN-WALL NANOWIRE BASED MAIN MEMORY

Compared with the conventional SRAM or DRAM by CMOS, the domain-wall nanowire based memory (DWM) can demonstrate two major advantages. Firstly, extremely high integration density can be achieved since multiple bits can be packed in one macro-cell. Secondly, zero standby power can be expected as a non-volatile device does not require to be powered to retain the stored data. In addition, the resistance-detection based readout does not require bit-line pre-charging, which avoids the sub-threshold leakage of the access transistors. In this section, we will present DWM-based design with macro-cell memory: structure, modeling, and data organization.

A. DWM macro-Cell design

Fig. 3: Macro-cell of DWM with: (a) single access-port; and (b) multiple access-ports

Figure 3(a) shows the design of domain-wall nanowire based memory (DWM) macro-cell with access transistors. The access-port lies in the middle of the nanowire, which divides the nanowire into two segments. The left-half segment of nanowire is used for data storage while the right-half segment is reserved for shift-operation in order to avoid information lost. In order to access the left-most bit, the reserved segment has to be at least as long as data segment. In such case, the data utilization rate is only 50%. In order to improve the data utilization rate, a multiple port macro-cell structure is presented in Figure 3(b). The access-ports are equally distributed along the nanowire, which divides the nanowire into multiple segments. Except the right-most segment, all other segments are data segments with the bits in one segment form a *group*. In such case, to access arbitrary

bit in the nanowire, the shift-offset is always less than the length of one segment, thus the data utilization rate is greatly improved.

Thus, the number of bits in one macro-cell can be calculated by

$$N_{cell-bits} = (N_{rw-ports} + 1)N_{group-bits} \quad (8)$$

In which $N_{rw-ports}$ is the number of access ports. Then the macro-cell area can be calculated by

$$A_{nanowire} = N_{cell-bits}L_{bit}W_{nanowire} \quad (9)$$

$$A_{cell} = A_{nanowire} + 2A_{shf-nmos} + 2A_{rw-nmos}N_{rw-ports} \quad (10)$$

where L_{bit} is the pitch size between two consecutive bits, $W_{nanowire}$ the width of domain-wall nanowire, $A_{shf-nmos}$ and $A_{rw-nmos}$ are the transistor size at shift-port and access-port respectively.

Moreover, the bit-line capacitance is crucial in the calculation of latency and dynamic power. The increased bit-line capacitance due to the multiple access-ports can be obtained by

$$C_{bit-line} = (N_{rw-ports}C_{drain-rw} + C_{drain-shf} + C_{bl-metal}) \times N_{row} \quad (11)$$

in which $C_{bl-metal}$ is the capacitance of bit-line metal wire per cell, the $C_{drain-rw}$ and $C_{drain-shf}$ are the access-port and shift-port transistor drain capacitances, respectively. Note that the undesired increase of per-cell capacitance will be suppressed by the reduced number of rows due to higher nanowire utilization rate.

Additionally, the domain-wall nanowire specific behaviors will incur in-cell delay and energy dissipation. The magnetization reversal energy $0.27pJ$ and delay $600ps$ can be obtained through the transient analysis by the SPICE-like simulation as discussed in Section III. The read-energy is in fJ scale thus can be omitted. Also, the read-operation will not contribute in-cell delay. The delay of shift-operation can be calculated by

$$T_{shift} = L_{bit}/v_{prop} \quad (12)$$

in which v_{prop} is the domain-wall propagation velocity that can be calculated by Equation 7. The Joule heat caused by the injected current is calculated as the shift-operation dynamic energy.

B. Cluster-group data organization

However, there are two potential problems for such DWM macro-cell. Firstly, there exists variable access latencies for the bits that locate at different positions in the nanowire. Secondly, if the required bits are all stored in the same nanowire, very long access latency will be incurred due to the sequential access.

It is important to note that the data exchange between main memory and cache is always in the unit of a cache-line size of data, i.e. the main memory will be read-accessed when last-level cache miss occurs; and will be write-accessed when a cache-line needs to be evicted. Therefore, instead of the per access latency, the latency of the data block in the size of a cache-line becomes the main concern. Based on such fact, we present a cluster-group based data organization. The idea behind *cluster* is to distribute data in different nanowires thus they can be accessed in parallel to avoid the sequential access; and the idea behind *group* is to discard the within-group addressing, and transfer the $N_{group-bits}$ bits in $N_{group-bits}$ consecutive cycles, to avoid the variable latency. Specifically, a cluster is the bundle of domain-wall nanowires that can be selected together through bit-line multiplexers. The number of nanowires in one cluster equals the I/O bus bandwidth of the memory. Note that the data in one cache-line have consecutive addresses. Thus, by distributing the bits of N consecutive bytes, where N is decided by the I/O bus bandwidth, into different nanowire within a cluster, the required N bytes can be accessed in parallel to avoid the sequential access. In addition, within each nanowire in the cluster, the data will be accessed

in the unit of group, i.e. the bits in each group will be accessed in consecutive cycles with a similar fashion as DRAM.

The number of groups per nanowire is thus decided by

$$N_{group-bits} = N_{line-bits} / N_{bus-bits}. \quad (13)$$

For example, in system with cache-line size of 64-byte, and memory I/O bus bandwidth of 64-bit, the group size is 8-bit. As such, the DWM with cluster-group based data organization can be operated in the following steps:

- Step1: The group-head initially is aligned with the access-port, thus the distributed first 8 consecutive bytes can be first transferred between memory and cache;
- Step2: Shift the nanowire with 1-bit offset, and transfer the following 8 consecutive bytes. Iterate this step 6 more times until the whole cache-line data is transferred;
- Step3: After the data transfer is completed, the group-head is relocated to the initial position as required in step 1.

As mentioned in Section III-C, the current-controlled domain-wall propagation velocity is proportional to the applied shift-current,. By applying a larger shift-current, a fast one-cycle cluster head relocation can be achieved. In such a manner, the data-transfer of cache block will be able to achieve a fixed and also lowest possible latency.

V. DOMAIN-WALL NANOWIRE BASED LOGIC

The magnetization switching with sub-nanosecond speed and sub-pJ energy have been experimentally demonstrated [9], [10], [11]. As such, the domain-wall nanowire based logic (DWL) can be further explored for logic-in-memory based computing. In this section, we show how to further build DWL based XOR-logic, and how it is applied for low-power ALU design for comparison and addition operations.

A. DWL-based XOR-logic Circuit

The GMR-effect can be interpreted as the bitwise-XOR operation of the magnetization directions of two thin magnetic layers, where the output is denoted by high or low resistance. In a GMR-based MTJ structure, however, the XOR-logic will fail as there is only one operand as variable since the magnetization in fixed layer is constant. Nevertheless, this problem can be overcome by the unique domain-wall shift-operation in the domain-wall nanowire device, which enables the possibility of DWL-based XOR-logic for computing.

Fig. 4: Low power XOR-logic implemented by two domain wall nanowires

A bitwise-XOR logic implemented by two domain-wall nanowires is shown in Figure 4. The proposed bitwise-XOR logic is performed by constructing a new read-only-port, where two free layers and one insulator layer are stacked. The two free layers are in the size of one magnetization domain and are from two respective nanowires. Thus, the two operands, denoted as the magnetization direction in free layer, can both be variables with values assigned through the MTJ of the according nanowire. As such, it can be shifted to the operating port such that the XOR-logic is performed.

For example, the $A \oplus B$ can be executed in the following steps

- The operands A and B are loaded into two nanowires by enabling WL_1 and WL_2 respectively;

- A and B are shifted from their access-ports to the read-only-ports by enabling SHF_1 and SHF_2 respectively;
- By enabling RD , the bitwise-XOR result can be obtained through the GMR-effect.

Note that in the x86 architecture processors, most XOR instructions also need a few cycles to load its operands before the logic is performed, unless the two operands are both in registers. As such, the proposed DWL-based XOR-logic can be a potential substitution of the CMOS-based XOR-logic. Moreover, similar as the DWM macro-cell, zero leakage can be achieved for such XOR-logic.

B. DWL-XOR based ALU

The DWL-XOR can be applied into ALU design in two function units, the N-input XOR and N-input full-adder for comparison and addition operations, respectively. As discussed in Section II, such two units account for more than half of the total transistors in ALU, and also are the most frequently visited units, especially in big data applications. The N-input XOR-logic can be realized in by employing N-bitwise DWL-XOR, which is able to take the highly intensive *comparison* instruction.

In the following, we present a full-adder design by DWL-XOR as shown in Figure 5. As each CMOS-based XOR-logic is built with

Fig. 5: Full adder design with DWL based XOR logic

four 2-input NAND gates, the substitution by DWL-based XOR-logic thereby can reduce roughly three quarters of the leakage power. Note that the addition of DWL-based full adder is also carried out in multiple cycles. Take $A + B + C_{in}$ as an example, the full-adder executes in the following steps.

- The operands A and B are loaded to the domain-wall nanowires in XOR1;
- The two operands are shifted to the read-only-port and the internal $A \oplus B$ result is generated;
- The CMOS-logic generates the C_{out} immediately;
- The internal $A \oplus B$ result and C_{in} are loaded into XOR2;
- The two operands are shifted to the read-only-port and the sum $S = A \oplus B \oplus C_{in}$ can be obtained.

As such, by connecting N full-adders together, an N-bit full-adder can be achieved and integrated into the ALU. Note that there are two more control-signals $ctrl1$ and $ctrl2$ used for nanowire operation control. A full-adder is able to execute both ADD and SUB instruction, thus together with the N-input DWL-XOR, a very large portion of the instructions can be optimized in terms of power reduction. In addition, the stalls caused by the slightly longer cycles can be greatly suppressed in the out-of-order superscalar processor.

VI. EXPERIMENT AND DISCUSSION

A. Experiment settings and methodology

We have the following evaluation of the proposed domain-wall nanowire based big-data computing platform with both memory and logic, respectively. The system configuration is shown in Table I. The 32-bit 65nm processor is assumed with four cores integrated. In each core, there are 6 integer ALUs which executes XOR, OR, AND, NOT, ADD and SUB operations, and complex integer operations like

MUL, DIV are executed in integer MUL. The 32nm technology node and 64-bit I/O bus width are assumed for memory.

Firstly, gem5 [19] simulator is employed to take both SPEC2000 and Phoenix benchmarks [20] and to generate the runtime instruction and memory accessing traces. The trace file is then analyzed with the statistics of instructions that can be executed on the proposed XOR and adder for logic evaluation. The L2-cache-miss rates are also generated, in order to obtain the actual memory access for memory evaluation.

Next, the McPAT[21] is extended with additional power models of DWL-based XOR-logic as described in Section V. As such, both accurate dynamic power and leakage power information of DWL-based ALU can be acquired. Meanwhile, as described in Section IV, the domain-wall nanowire based memory differs from the conventional CMOS-based memory in many aspects, thus CACTI[22] has been extended with the domain-wall nanowire based memory model for DWM based main memory, with accurate device operation energy and delay data obtained from SPICE simulator developed in Section III.

TABLE I: System configuration

Processor	
Technology node	65nm
Number of cores	4
Frequency	1GHz
Architecture	x86, O3, issue width - 4, 32 bits
Functional units	Integer ALU - 6 Complex ALU - 1 Floating point unit - 2
Cache	L1: 32KB - 8 way/32KB - 8 way L2: 1MB - 8 way Line size - 64 bytes
Memory	
Technology node	32nm
Memory size	2GB - 128MB per bank
IO bus width	64 bits

B. Logic-in-memory architecture

The transient analysis of the domain-wall nanowire XOR structure has been performed in the SPICE simulator. with both controlling timing diagram and operation details shown in Figure 6.

Fig. 6: The timing diagram of DWL-XOR with SPICE-level simulation for each operation

The current density of $7e10A/m^2$ is utilized for magnetization switching. The θ states of the nanowire that takes A are all initialized at 0, and the one takes B all at π . Only two-bit per nanowire is assumed for both nanowires. The operating-port is implemented as a developed magnetization controlled magnetization (MCM) device, with internal state variables θ and ϕ for both upper layer and bottom layer. In the cycles of *loadA* and *loadB*, the precession switching

can be observed for the MTJs of both nanowires. Also, the switching energy and time have been calculated as 0.27pJ and 600ps, which is consistent with the reported devices [9], [10], [11]. In the *shift* cycles, triggered by the *SHF*-control signal, the dynamics θ and ϕ of both upper and bottom layers are updated immediately. In the *operation* cycle, a subtle sensing current is applied to provoke GMR-effect. Subtle magnetization disturbance is also observed in both layers in the MCM device, which validates the read-operation. The θ values that differ from initial values in the *operation* cycle also validate the successful domain-wall shift.

C. Power evaluation of main memory

TABLE II: Performance comparison of 128MB memory-bank implemented by different structures

Memory structure	area (mm^2)	access energy (nJ)	access time (ns)	leakage (mW)
DRAM	20.5	0.77	3.46	620.2
DWM/1 port	8.9	0.65	1.90	48.4
DWM/2 ports	6.2	0.72	1.71	30.1
DWM/4 ports	6.2	0.89	1.69	24.3
DWM/8 ports	5.7	1.31	1.88	19.0

Table II shows the 128MB memory-bank comparison between CMOS-based memory (or DRAM) and domain-wall nanowire based memory (or DWM). The number of access ports in main memory is varied for design exploration. The results of DRAM are generated by configuring the original CACTI with 32nm technology node, 64-bit of I/O bus width with leakage optimized. The results of the DWM are obtained by the modified CACTI according to Section IV with the same configuration.

It can be observed that the memory area is greatly reduced in the DWM designs. Specifically, the DWMs with 1/2/4/8 access ports can achieve the area saving of 57%,70%,70% and 72%, respectively. The trend also indicates that the increase of number of access-ports will lead to higher area saving. This is because of the higher nanowire utilization rate, and is consistent with the analysis discussed in Section IV. Note that the area saving in turn results in a smaller access latency, and hence the DWM designs on average provide 1.9X improvement on the access latency. However, the DWM needs one more cycle to perform shift operation, which will cancel out the latency advantage. Overall, the DWM and DRAM have similar speed performance. In terms of power, the DWM designs also exhibit benefit with significantly leakage power reduction. The designs with 1/2/4/8 access ports can achieve 92%,95%,96% and 97% leakage power reduction rates, respectively. The advantage mainly comes from the non-volatility of domain-wall nanowire based memory cells. The reduction in area and decoding peripheral circuits can further help leakage power reduction in DWM designs. In addition, the DWM designs have the following trend of access energy when increasing the number of access ports. The designs with 1/2 ports require 16% and 6% less energy, while designs with 4/8 ports incur 15% and 70% higher access energy cost. This is because when the number of ports increases, there are more transistors connected to the bit-line which leads to increased bit-line capacitance.

Fig. 7: (a) the runtime dynamic power of both DRAM and DWM under Phoenix and SPEC2006 (b) the normalized intended memory accesses

The runtime dynamic power comparison under different benchmark programs are shown in Figure 7(a). It can be seen that the dynamic power is very sensitive to the input benchmark, and the results of the Phoenix benchmarks shows no significant difference from those in SPEC2006. This is because the dynamic power is effected by both intended memory access frequency and the cache miss rate. Figure 7(b) shows the normalized intended memory reference rate, and as expected the data-driven Phoenix benchmarks have several times higher intended memory reference rate. However, both L1 and L2 cache miss rates of Phoenix benchmarks are much lower than SPEC2006, which is due to the very predictable memory access pattern when exhaustively handling data in Phoenix benchmarks. Overall, the low cache miss rates of Phoenix benchmarks cancel out the higher memory reference demands, which leads to a modest dynamic power. Also, the runtime dynamic power contributes much less to the total power consumption compared to leakage power, thus the leakage reduction should be the main design objective when determining the number of access ports.

D. Power evaluation of ALU logic

From DWL-XOR based ALU design evaluation, McPAT is modified to evaluate power of the 32-bit ALU that is able to perform XOR, OR, AND, NOT, ADD and SUB operations. The instruction controlling decoder circuit is also considered during the power evaluation. The leakage power of both designs is calculated at gate level by the McPAT power model.

Fig. 8: The per core ALU power comparison between CMOS design and DWL based design

Figure 8 presents the per-core ALU power comparison between the conventional CMOS design and domain-wall nanowire logic (DWL) based design. Benefited from the use of DWL, both of the dynamic power and leakage power can be greatly reduced. It can be observed that the set of Phoenix benchmarks consume higher dynamic power compared to those of SPEC2006, which is due to the high parallelism of MapReduce framework with high utilization rate of the ALUs. Among each set, the power results exhibit a low sensitivity to the input, which indicates that percentages of instructions executed in XOR and ADDER of ALU are relatively stable even for different benchmarks. The stable improvement ensures the extension of the proposed DWL to other applications. Averagely, a dynamic power reduction of 31% and leakage power reduction of 65% can be achieved for ALU logic based on all the eight benchmarks.

VII. CONCLUSION

The domain-wall nanowire based memory-computing is explored in this paper towards big-data applications such as web-searching. The shift-operation in domain-wall nanowire has been adapted to perform logic operations such as XOR-logic for comparison as well

as addition. Therefore, one can build a highly logic-in-memory computing platform with both memory and logic implemented by domain-wall nanowire devices. The SPICE behavior model of domain-wall nanowire is also developed for both circuit-level verification and system-level evaluation. Utilizing applications of general-purpose SPEC2006 benchmark and also web-searching oriented Phoenix benchmark, we find that the domain-wall nanowire based computing can reduce 92% leakage power and 16% dynamic power when compared with main memory implemented by DRAM; and can also reduce 31% both dynamic and 65% leakage power under the similar performance when compared with ALU by implemented by their CMOS counterpart.

ACKNOWLEDGMENT

This work is sponsored by Singapore MOE TIER-2 fund MOE2010-T2-2-037 (ARC 5/11) and A*STAR SERC-PSF fund 11201202015.

REFERENCES

- [1] J. Lin and C. Dyer, "Data-intensive text processing with mapreduce," *Synthesis Lectures on Human Language Technologies*, 2010.
- [2] H. Wong and et.al., "Phase change memory," *Proceedings of the IEEE*, 2010.
- [3] B. C. Lee and et.al., "Phase-change technology and the future of main memory," *Micro, IEEE*, 2010.
- [4] K. Tsuchida and et.al., "A 64mb mram with clamped-reference and adequate-reference schemes," in *Proc. ISSCC*, 2010.
- [5] D. Halupka and et.al., "Negative-resistance read and write schemes for stt-mram in 0.13 um cmos," in *Proc. ISSCC*, 2010.
- [6] Y. Wang and H. Yu, "Design exploration of ultra-low power non-volatile memory based on topological insulator," in *Proc. NANOARCH*, 2012.
- [7] D. B. Strukov and et.al., "The missing memristor found," *Nature*, 2008.
- [8] Y. Wang and et.al., "Design of low power 3d hybrid memory by non-volatile cbam-crossbar with block-level data-retention," in *Proc. ISLPED*, 2012.
- [9] H. Zhao and et.al., "Sub-200 ps spin transfer torque switching in in-plane magnetic tunnel junctions with interface perpendicular anisotropy," *Journal of Physics D: Applied Physics*, 2011.
- [10] G. Rowlands and et.al., "Deep subnanosecond spin torque switching in magnetic tunnel junctions with combined in-plane and perpendicular polarizers," *Applied Physics Letters*, 2011.
- [11] H. Zhao and et.al., "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*, 2011.
- [12] S. S. Parkin and et.al., "Magnetic domain-wall racetrack memory," *Science*, 2008.
- [13] L. Thomas and et.al., "Racetrack memory: A high-performance, low-cost, non-volatile memory based on magnetic domain walls," in *Proc. IEDM*, 2011.
- [14] R. Venkatesan and et.al., "Tapecache: A high density, energy efficient cache based on domain wall memory," in *Proc. ISLPED*, 2012.
- [15] M. Stiles and J. Miltat, "Spin-transfer torque and dynamics," Springer Berlin / Heidelberg, 2006.
- [16] D. Chiba and et.al., "Control of multiple magnetic domain walls by current in a co/nl nano-wire," *Applied Physics Express*, 2010.
- [17] W. Fei and et.al., "Design exploration of hybrid cmos and memristor circuit by new modified nodal analysis," *IEEE TVLSI*, 2011.
- [18] Y. Shang and et.al., "Analysis and modeling of internal state variables for dynamic effects of nonvolatile memory devices," *IEEE TCAS-I*, 2012.
- [19] The gem5 simulator. [Online]. Available: <http://www.m5sim.org>
- [20] C. Ranger and et.al., "Evaluating mapreduce for multi-core and multi-processor systems," in *Proc. HPCA*, 2007.
- [21] S. Li and et.al., "Mcpat: an integrated power, area, and timing modeling framework for multicore and manycore architectures," in *Proc. MICRO*, 2009.
- [22] Cacti: An integrated cache and memory access time, cycle time, area, leakage, and dynamic power model. [Online]. Available: <http://www.hpl.hp.com/research/cacti/>