DW-AES: A Domain-Wall Nanowire-Based AES for High Throughput and Energy-Efficient Data Encryption in Non-Volatile Memory

Wang, Yuhao; Ni, Leibin; Chang, Chip-Hong; Yu, Hao

2016

Wang, Y., Ni, L., Chang, C.-H., & Yu, H. (2016). DW-AES: A Domain-Wall Nanowire-Based AES for High Throughput and Energy-Efficient Data Encryption in Non-Volatile Memory. IEEE Transactions on Information Forensics and Security, 11(11), 2426-2440.

<https://hdl.handle.net/10356/85627>

<https://doi.org/10.1109/TIFS.2016.2576903>

© 2016 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works. The published version is available at: [<http://dx.doi.org/10.1109/TIFS.2016.2576903>].

Downloaded on 26 Apr 2023 23:09:54 SGT

DW-AES: A Domain-wall Nanowire Based AES for High Throughput and Energy-efficient Data Encryption in Non-volatile Memory

Yuhao Wang, Leibin Ni, Chip-Hong Chang, *Senior Member, IEEE*, and Hao Yu, *Senior Member, IEEE*

Abstract—Big-data storage poses significant challenges to anonymization of sensitive information against data sniffing. Not only will the encryption bandwidth be limited by the I/O traffic, the transfer of data between processor and memory will also expose the input-output mapping of intermediate computations on I/O channels that are susceptible to semi-invasive and non-invasive attacks. Limited by the simplistic cell-level logic, existing logic-in-memory computing architectures are incapable of performing the complete encryption process within the memory at reasonable throughput and energy efficiency. In this paper, a block-level in-memory architecture for Advanced Encryption Standard (AES) is proposed. The proposed technique, called DW-AES, maps all AES operations directly to the domain-wall nanowires. The entire encryption process can be completed within a homogeneous, high-density and standby-power-free non-volatile spintronic based memory array without exposing the intermediate results to external I/O interface. Domain-wall nanowires based pipelining and multi-issue pipelining methods are also proposed to increase the throughput of the baseline DW-AES with insignificant area overhead and negligible difference on leakage power and energy consumption. The experimental results show that DW-AES can reduce the leakage power and area by orders of magnitude compared to existing CMOS ASIC accelerators. It has an energy efficiency of 22 pJ/bit, which is $5\times$ and $3\times$ better than the CMOS ASIC and memristive CMOL based implementations, respectively. Under the same area budget, the proposed DW-AES achieves $4.6\times$ higher throughput than the latest CMOS ASIC AES with similar power consumption. The throughput improvement increases to $11\times$ for pipelined DW-AES at the expense of doubling the power consumption.

I. INTRODUCTION

Emerging nano-scale memories such as ReRAM [1] and STT-RAM [2], [3] bring instant power-up and ultra-low leakage power non-volatility into the CMOS circuits for big-data storage. With data retains during reboot or suspension, these non-volatile memories (NVMs) are also vulnerable to data sniffing attacks. Data sniffing is one of the main causes of mass break-ins on the internet today. If the information in a networking environment is passed along the Ethernet in clear-text, hackers can easily gain root to add a promiscuous node that could compromise all other nodes and storages in the net. One solution to combat data sniffing is data anonymization. Unfortunately, it takes several hours to encrypt/decrypt one Terabyte of data in 128-bit Advance Encryption Standard (AES) on a general purpose computer. While cheap distributed storage is the driving factor behind big data, the cost for keeping its data confidentiality can be intimidating. Limited by footprint and energy dissipation, data encryption is less

affordable on resource constrained devices. In this regard, area-efficient, low-power and high throughput in-memory encryption engine is of great significance not only for privacy in big data processing, but also on resource-constrained environments.

Various CMOS-based hardware implementations for AES have been proposed [4], [5], [6]. However, CMOS-based ASIC implementations tend to dissipate high static power in current deep sub-micron technology with their speed limited by low-voltage operation. In [7], a memristive CMOL implementation based on hybrid CMOS and ReRAM is introduced to improve the performance of AES encryption. As the storage device is non-volatile, leakage current is cut-off completely when unused. However, the ReRAM serves primarily as reconfigurable interconnection. Although the distance between memory and logic circuits has been reduced in the hybrid design, the logic and memory are still separated.

In the abovementioned classical and hybrid memory-logic architectures, data stored in the NVM needs to be loaded into the main memory for encryption, and then written back to the NVM. Large static power are dissipated in on-chip memory as the power supply must be continuously applied in the volatile memory. Due to the Von-Neumann bottleneck, more power is expended on reading/writing the data than on performing the encryption operations. As wire delay dominates chip performance at fabrication node below 90nm, leakage current and data traffic become more critical. To overcome the bandwidth limitation of I/O traffic, cell-level logic-in-memory architecture is explored in [8], [9]. The basic idea is to have the data preprocessed in the memory so that the processor is provided with only the intermediate results instead of a large volume of raw data. As the in-memory logic remains idle during normal memory accesses and is only active upon request, leakage power is also reduced significantly. The main constraint of cell-level in-memory logic is the scalability and endurance of memory devices to accommodate sophisticated operations required by the application.

Recent experiments on spintronic devices have demonstrated their great scalability, infinite endurance and high switching/sensing speed [3], [10], [11]. As nano-magnets can be vertically stacked above the CMOS circuits at the back-end process, 3D integration of Spin Transfer Torque Magnetic RAM (STT-MRAM) on CMOS logic circuit has been fabricated to boost computing speed and minimize data traffic power [12]. Beyond high-density and standby-power-free memory [13], spintronic is also a promising technology for big-data storage with logic-in-memory computing capability. Such capability was first exploited in our preliminary design of DW-AES [14] based on the emerging domain-wall (DW) nanowire [15], [16], [17]. In this paper, the key differentiation

Y. Wang, L. Ni, C.H. Chang and H. Yu are all with School of Electrical and Electronic Engineering, Nanyang Technological University (NTU), Singapore. This work is sponsored by Singapore NRF-CRP (NRF-CRP9-2011-01) and MOE Tier-2 (MOE2010-T2-2-037 (ARC 5/11)). Please address comments to haoyu@ntu.edu.sg.

of block-level from cell-level in-memory computing architecture that enables the full mapping of AES operations by DW nanowire devices is elaborated for the first time. Supported by a more comprehensive DW nanowire device modelling for system-level simulation, optimization algorithms for mapping the AES operations into the original, pipeline and multi-issue pipelined block-level DW nanowire architectures are proposed. The two newly proposed pipelined designs have overcome the highly asymmetric latencies of different AES operations in our preliminary DW-AES design [14], and led to significantly improved throughput over the state-of-the-art CMOS AES [4] published after it.

Specifically, the block-level in-memory encryption offers two major advantages over the existing approaches. Firstly, all AES ciphers can be computed in-place and merged into the memory with no restriction on the pairing of operand bits. Hence the block cipher can be easily chained and multiple encryptions can be independently performed on the targeted data that is stored in non-volatile DW memory. Secondly, the DW-AES cipher is implemented homogeneously by the DW nanowire devices with magnetic tunnel junction (MTJ) sensing. This provides a tight integration between DW-AES ciphers and memory elements, and allows the peripheral circuits like decoders and sense amplifiers (SAs) to be reused. Not only does the chip area reduce tremendously, but the global wire count also cuts down substantially, leading to lower wire delay and dynamic power dissipation. The proposed DW-AES enjoys the inherent standby-power-free property and nearly infinite endurance of spintronic devices, which are particularly important for logic-in-memory architecture as more devices in the memory array are idle most of the time, and more frequent internal accesses to memory data are required due to the merging of logic and memory. By virtue of the shift-register like domain wall movement, the scale and complexity of AES operations implementable by the proposed block-level in-memory computing architecture is unrivalled by other cell-level in-memory AES computing architectures with the same power budget. The proposed pipelined and multi-issue pipelined DW-AES architectures further increase the throughput of DW-AES. Our experimental results show that the proposed pipelined DW-AES outperforms the CMOS-based ASIC and hybrid CMOS-ReRAM for AES computing in big-data storage, with $11\times$ and $4.4\times$ throughput improvements and $5\times$ and $3\times$ higher energy efficiency, respectively.

The rest of the paper is organized as follows. Section II introduces the background of AES algorithm and logic-in-memory architecture. Section III discusses the proposed non-volatile in-memory architecture for AES computing with DW nanowire device modeling. Section IV details how AES transformations can be mapped by the DW nanowire technology. Section V presents two variants of pipelined in-memory AES computing. Experiment results are presented in Section VI and the paper is concluded in Section VII.

II. BACKGROUND

Algorithm 1: Advanced Encryption Standard

Input: 128-bit plaintext

- 1: Organize input data as 4×4 state matrix M_s , each entry $S_{i,j}$ has one byte
- 2: **for** $r = 1 : N_r$ **do**
- 3: **for all** $S_{i,j} \in M_s$ **do**
- 4: SubBytes transformation: $S'_{i,j} \leftarrow f(S_{i,j})$
- 5: **end for**
- 6: **for all** $S_{n,j} \in M_s$ in n_{th} row, $n \in \{1, 2, 3, 4\}$ **do**
- 7: $k \leftarrow (j-n+1) \bmod 4$
- 8: ShiftRows transformation: $S'_{n,j} \leftarrow S_{n,k}$, i.e. each row is left shifted circularly by $n-1$ bytes
- 9: **end for**
- 10: **if** $i \neq N_r$ **then**
- 11: MixColumns transformation: $M_s \leftarrow M_s \times M_{mc}$
- 12: **end if**
- 13: **for all** $S_{i,j} \in M_s$ **do**
- 14: AddRoundKey transformation: $S'_{i,j} \leftarrow S_{i,j} \oplus K_{i,j}$
- 15: **end for**
- 16: $M_s \leftarrow M'_s$
- 17: **end for**

Output: 128-bit ciphertext

A. Preliminary on Advanced Encryption Standard

The Advanced Encryption Standard (AES) algorithm operates on plaintext block of 128 bits. The 16-byte input data is internally organized into an array of four rows by four columns, called the *state matrix* (M_s). The input data is encrypted by applying a sequence of transformations to the state matrix, as detailed by the pseudo code in Algorithm 1. The flow chart in Figure 1 shows the dominant resources used for the hardware implementation of each transformation module, where the gate utilization data is obtained by synthesizing the AES Verilog code from [18]. The operations performed and the resources consumed for each module are described as follows.

- SubBytes: Each byte $S_{i,j}$ of the state matrix M_s will be independently updated by a nonlinear transformation f in this module. The mapping f is performed by a substitution-box (S-box), which takes one byte of input from M_s and transforms it into another byte at the same

Fig. 1. Flow chart of AES algorithm with gate utilization analysis

position. The SubBytes module accounts for half of the total gates in AES, with registers used as fixed storage elements of the look-up table (LUT). Each S-box is pre-configured with an 8-bit word in each memory location addressable by an 8-bit input. Hence the LUT size is $2^8 \cdot 8 = 2048$ bits. The percentage of hardware resources utilized by this module may vary depending on how the S-box is implemented. If the S-box is implemented by combinational logic circuit, XOR gates become the dominant resources, which account for more than 70% of gate utilization for the AES implementation, as reported in [7].

- **ShiftRows:** The n_{th} row of M_s will be cyclically shifted to the left by n bytes. As shown by arrows on the entries of M_s in Figure 1, the top row is not shifted; the second row is shifted by one byte position; the third row by two; and the fourth row by three. In ASIC design, the ShiftRows transformation can be performed in-place by storing the content of M_s in shift registers, hence no additional logic gate is incurred.
- **MixColumns:** Each column of M_s is multiplied by a constant matrix M_{mc} consisting of three integer values 1, 2 and 3, as shown in Figure 1. The multiplication of a variable byte by an integer 1, 2 or 3 of M_{mc} results in an unchanged byte, a 1-bit left shifted byte or the XOR of the byte with a 1-bit left shifted version of itself, respectively. Hence, this invertible linear transformation replaces all four bytes in a column of M_s such that each byte is mixed with all four bytes in the column. This module contains only XOR gates and it accounts for nearly half of the total number of gates.
- **AddRoundKey:** The 16-byte round keys are organized in a similar 4×4 array M_k as the state matrix. Each entry of M_k is denoted as $K_{i,j}$. In this operation, each byte $S_{i,j}$ of M_s will be replaced by the result of a bitwise-XOR operation with a byte $K_{i,j}$ in the same row and column of the round key matrix M_k . Therefore, the AddRoundKey module is again built by merely XOR gates, which accounts for 3.3% of the total gates.

In conclusion, the basic operations involved in AES are XOR, shift, and table look up although the percentage utilization of the logic gates used for each module may vary from design to design.

B. Non-volatile Logic-in-memory Architecture

Conventionally, all the data is kept in the memory separated from the processor but connected with the I/Os. During the execution, all data needs to be migrated to the processor and written back thereafter. This will cause I/O congestion and impact the system performance for data-intensive application. In addition, significant standby power will be consumed in holding the large volume of data.

Theoretically, the bandwidth problem can be overcome by adding more I/Os operating at higher frequency. However, this solution is limited by the scalability of CMOS technology and has non-trivial cost penalty. Alternatively, the required data communication traffic between memory and processor

Fig. 2. Conventional in-memory computing architecture at memory cell level [8]

can be reduced. Instead of feeding the processor with a large volume of raw data, the data can be preprocessed so that the processor needs to deal only with a smaller amount of intermediate results, thereby reducing the communication traffic. The *logic-in-memory* architecture merges logic with memory to allow some preprocessing to be done locally. Taking the leakage reduction into consideration, the logic-in-memory architectures associated with NVM are presented in [8], [9]. Figure 2 shows the cell-level circuit for the logic-in-memory architecture of [8]. The logic function realized in this example circuit is an in-memory full-adder with both *sum* logic and *carry* logic. Instead of reading out the operands, the *sum* and *carry* results can be obtained directly after the read operation, as indicated by the red arrow.

As the logic is inserted into one cell or a few cells, it is limited to only simple logic or else the overhead would be impractically large. Although this may offload the processor to some extent, its effectiveness in reducing the operands and hence the communication traffic is limited. It also suffers from similar shortcoming as memory array in that only very few logic cells are active at any one time. As majority of the in-memory logic circuits are idle most of the time, it means a waste of computational resources and additional leakage power dissipated by the unused logic. Another disadvantage is the organization of in-memory logic imposes a very strict ordering requirement on the data to be stored. For example, each pair of bits from the same weighted bit position of the integer operands, A and B , for $A \oplus B$ needs to be stored in the same pair of memory cells with an in-memory XOR logic cell. If another calculation of $A \oplus C$ is required, then each bit of A needs to be duplicated in order to pair with the corresponding bit of operand C in another pair of memory cells.

Although the data transmission between memory and processor has been substantially reduced by the in-memory architecture, there is still a need to interface with the processor for the transmission and execution of new instructions dedicated to in-memory computing. In the proposed architecture, we have adopted the control bus and communication protocol described in [19] so that very few data transmissions between processor and memory are required for the in-memory AES.

Fig. 3. The overview of in-memory AES computing architecture at memory block level

III. PROPOSED BLOCK-LEVEL NON-VOLATILE IN-MEMORY COMPUTING ARCHITECTURE WITH DOMAIN-WALL NANOWIRE CELLS

A. Proposed Block-level In-memory Computing Architecture

In view of the deficiencies of existing cell-level in-memory architecture, an alternative in-memory architecture is proposed, where the logics for in-memory computing are distributed at block level. An overview of the proposed in-memory architecture is illustrated in Figure 3. The communication between processor and memory is organized in a H-tree network. The data block can be the data array or a number of data arrays that belong to the same branch of ‘H-tree’. Instead of inserting the in-memory logic at memory cell level inside the data array, the architecture in Figure 3 pairs each block of data with in-memory logic. This architecture offers two main advantages to overcome the operand reduction bottleneck of cell-level in-memory architecture. Firstly, higher complexity in-memory logic can be accommodated at block level than at cell level. The ratio between the logic and data storage resources can also be tailored to meet the application requirements. A higher logic/storage ratio offers higher execution throughput at the expense of lower storage efficiency, and vice versa. Such flexibility will therefore allow better trade-off between execution parallelism and storage efficiency. Secondly, data flow between logic and memory is highly localized to within the leaves of the H-tree. Using AES as example, the data flow of the block-level in-memory architecture involves reading the plaintext data by each in-memory logic block from its corresponding paired data block, performing the AES operations, and writing back the encrypted results to the same data array. As the encryption of every input block is independent in AES, all in-memory blocks can execute independently. So only local data transmission is required, which eliminates the need for communications among in-memory logic blocks. By avoiding the complication of data dependency, memory access control can be greatly simplified.

Two other problems, which are the area and leakage power

overheads of CMOS logic mentioned in Section II-B, remain to be addressed by the block-level architecture. Even through block-level architecture can accommodate more in-memory logic, larger memory area overhead will complicate the routing of H-tree and increases the access time of memory as a whole. In addition, longer H-tree wire will have larger parasitic and consume more energy per access. Since the in-memory logic is only active on demand, the entire in-memory architecture is essentially a data storage bank that consumes predominantly standby power. The solution to these two problems calls for a compact and ultra-low leakage device technology. From the *sum* and *carry* logic circuit realized in [10], the non-volatile DW nanowire is found to be the most suitable device for implementing the in-memory logic of the proposed block-level architecture. DW nanowire [15], [16], [17] is itself a non-volatile leakage-free memory device as it uses spin rather than charge as the physical state variable for storing information. As will be illustrated in later section, the inherent match between the characteristics of DW nanowire and the key logic operations of AES has drastically reduced the circuit complexity. Together with its feasibility to operate at ultra-low voltage, both storage and logic functions can be accomplished with high energy efficiency by integrating DW devices into the proposed in-memory architecture. In contrast to a CMOS transistor, multiple bits of information can be stored in a single ferromagnetic nanowire. The remarkably high storage efficiency of DW nanowire can be used to trade for greater execution parallelism and I/O reduction by the in-memory logic in block-level architecture. In what follows, the fundamentals of DW nanowire and its modelling will be presented.

B. Domain-wall Nanowire Device Modeling

A DW nanowire consists of multiple magnetic domains that are separated by non-magnetic regions called domain walls along a nano-magnetic strip. The basic DW nanowire structure is shown in Figure 4(a). The spin-polarity (i.e., leftward or rightward magnetization) of a domain can be used to represent a binary bit. The spin-polarity in each domain can be sensed by placing below it a strongly magnetized ferromagnetic layer separated by a thin tunneling oxide barrier. Such a sandwich-like structure forms a magnetic-tunnel-junction (MTJ), through which the stored information can be accessed. 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 stored information in each domain separated by the domain walls is preserved. Such a tape-like operation will shift all the bits simultaneously in one direction like a shift register.

Writing a target domain of the nanowire can be modeled as the magnetization reversal of the free layer of its MTJ. 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 4(b). The dynamics of spin-current induced magnetization can be described by the expressions of θ and ϕ given in [20]

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

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

where θ_0 is the initial value of θ , which is slightly tilted from the stable x or $-x$ directions; t_0 is the precession time constant; ω is the angular speed of ϕ ; k_1 to k_4 are the magnetic parameters detailed in [20]; and I is the spin-current that causes the magnetization precession.

As described earlier, the state of a storage domain can be read by an MTJ. A typical R-V characteristic of MTJ is shown in Figure 4(c). It consists of a giant magnetoresistance (GMR) region and a tunneling region. Depending on the alignment of magnetization directions of the fixed and free layers, an MTJ can assume either the low resistance state, R_l if the magnetizations of both layers are parallel or the high resistance state R_h if they are anti-parallel. As such, the MTJ resistance can be calculated by the 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 the free (upper) layer and the fixed (bottom) layer, R_{l0} and R_{h0} are the low and high MTJ resistances under an applied read voltage as shown in Fig. 4(c).

The movement of domain wall in a nanowire can be digitalized and modeled in the unit of domains. This makes it analogous to a shift register operation as one bit is stored in each domain. Except the bit in the MTJ, all other bits represented 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 its adjacent domains. This implies that the shifting of domain wall can be modeled as a magnetization controlled magnetization (MCM) device, which is similar to traditional controlled sources but with magnetization state variables θ and ϕ as controlled variables.

The DW propagation velocity can be mimicked by a voltage controlled oscillator. Higher shift voltage/current will lead to more frequent MCM propagations, i.e., higher propagation velocity. The link between shift current and propagation velocity

Fig. 4. (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 of MTJ

is experimentally observed by its current-velocity relation [21] as follows:

$$v = k(J - J_0), \quad (4)$$

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

By combining Equations (1) to (4), with the magnetization angles θ and ϕ as the internal state variables, the behaviors of the DW nanowire device can be fully described by modeling each domain as a MCM device. The modified nodal analysis can be built in the SPICE-like simulator [22]. As the MTJ junction model and the shift behaviour have been validated in [23] and [21], respectively, accurate device-level simulation can be performed to evaluate the performance of DW nanowire based in-memory architecture for AES operations.

IV. DOMAIN-WALL NANOWIRE BASED AES COMPUTING

In this section, the detailed DW nanowire based in-memory encryption will be described and discussed.

A. Data Organization of State Matrix

Because in-memory encryption is performed directly on the data cells, the data needs to be organized in certain fashion to facilitate the execution of AES algorithm. As discussed in Section III-B, DW nanowires can only support serial access, i.e., one bit of information can be accessed from a DW nanowire at any one time. In order to access multiple bits within one cycle, the data needs to be distributed into separate nanowires so that they can be operated concurrently. In AES algorithm, the basic processing unit is one byte of the state matrix M_s . Therefore, M_s is split into eight 4×4 binary arrays $A_k \forall k \in [0, 7]$, as illustrated in Figure 5. The binary bit in the i -th row and j -th column of each array A_k corresponds to the k -th bit of the state byte $S_{i,j}$ of M_s . By distributing the Boolean operations on state bytes $S_{i,j}$ into eight binary arrays, the byte access requirement of AES algorithm is satisfied.

To exploit the shift property of DW nanowire for the ShiftRows transformation of AES, each row of the binary array needs to be stored in one DW nanowire. Thus, four nanowires is needed for each array $A_k, k \in [0, 7]$. In each nanowire, some redundant bits are appended to the four data bits to reduce

Fig. 5. Distributed data organization of state matrix for in-memory operation by domain-wall nanowire devices

Fig. 6. SubBytes stage with S-box function realized by domain-wall memory based look-up table

the shift current required for the circular left shift operation. The rationale will be elaborated later in the implementation of ShiftRows transformation.

B. SubBytes

Instead of using SRAM LUT to implement the S-box, we propose to implement the S-box by non-volatile DW nanowire devices, abbreviated as DW-LUT, as shown in Figure 6. **The 8-bit results are split into separated nanowires to speed up this nonlinear transformation.** Eight SAs are required for each DW-LUT in order to output the result parallelly. As both the in-memory state matrix and S-box are made up of non-volatile DW nanowires, leakage power is reduced significantly. Further area and power savings are achieved by sharing the decoders and SAs for the memory and DW-LUT. The number of cycles consumed in the *SubBytes* stage, t_{SB} , can be derived as follows:

$$t_{SB} = (t_{read} + t_{LUT} + t_{write}) \cdot \frac{16}{N_{LUT}} \quad (5)$$

where $N_{LUT} \in \{1, 2, 4\}$, t_{read} and t_{write} are the read and write latency (in number of clock cycles) of state matrix M_s , and t_{LUT} is the latency of look-up table in number of cycles.

As each DW-LUT handles one $S_{i,j}$ transformation at a time, the SubBytes transformations can be parallelized by using more DW-LUT. The maximum value of N_{LUT} is limited by the number of bytes that can be accessed in the state matrix M_s . As each A_k can have at most four SAs, one to each nanowire for each row of A_k , at most four bytes can be accessed simultaneously. The SAs can be shared among different nanowires to save area if full parallelism is not required.

The composite fields for S-box and inverse S-box operations can be represented using normal basis, polynomial basis or mixed basis [24]. In the proposed DW nanowire architecture, DW-LUT instead of combinational logic is used to store the transformed data of SubBytes and InvSubBytes, hence the hardware complexity of their implementation is not affected by using different bases for the mapping to composite field S-boxes.

C. ShiftRows

As opposed to the conventional computing flow, the ShiftRows transformation is performed directly on the stored data in an in-memory computing manner with the unique shift property of DW nanowire. Due to the distributed data organization, the ShiftRows transformation is performed by circularly left shifting the binary bits in the nanowires corresponding to the i -th rows of all A_k by i bits $\forall i \in \{0, 1, 2, 3\}$. In other words, the second row of each bit matrix A_k needs to be left shifted cyclically by one bit, the third row by two bits, and the fourth row by three bits, while the top row remains unchanged. To avoid writing back the most significant bit (MSB) to the least significant bit (LSB) for each circular left shift operation, redundant bits are padded to form a virtual circle on the nanowire, as illustrated in Figure 7.

The number of redundant bits (domains) to be added in each nanowire is determined by the number of bits to be circularly left shifted for each row. As the shift operation for each nanowire can be performed concurrently, the number of cycles consumed in the *ShiftRows* stage, t_{SR} , can be determined by the longest row shift delay, i.e.,

$$t_{SR} = \max(t_{rs0}, t_{rs1}, t_{rs2}, t_{rs3}) \quad (6)$$

where t_{rs_i} denotes the row shift delay of the i -th row, $i \in \{0, 1, 2, 3\}$.

In order to complete all shifts in one cycle, shift current of different amplitudes is applied to each row according to the linear current-velocity relationship of shift operation [25]. This implies that the amplitudes of the shift currents applied to the nanowires for the third and fourth rows are twice and three times more than that applied to the nanowire for the second row. To reduce the shift current, consider the following equivalent circular left and right shift operations of a four-bit operand: $LS(1) \stackrel{\text{def}}{=} RS(3)$, $LS(2) \stackrel{\text{def}}{=} RS(2)$, $LS(3) \stackrel{\text{def}}{=} RS(1)$, where $LS(i)$ and $RS(i)$ denote the left and right shift operations with the number of bits to be shifted indicated by the integer argument i . This means that instead of shifting three bits leftward for the last row, only one bit right shifting is required, which reduces the redundant data from three bits to one bit and the amplitude of the applied shift current to only one third of what is originally required. In Figure 7, the bits in the same color are synchronized bits. To ensure that the

Fig. 7. ShiftRows transformation by domain-wall nanowire shift operations

Fig. 8. AddRoundKey stage with XOR logic realized by domain-wall nanowire

correct results will always be obtained by the circular shift, the redundant bits must also be updated coherently as the state matrix changes.

D. AddRoundKey

In the AddRoundKey stage, each byte in the state array will be bit-wise XORed with the corresponding key byte. As the key logic operation in this stage is XOR, we propose a nanowire based XOR logic (DW-XOR) for leakage free computing. As described in Section III-B, the GMR-effect can be considered as an XOR operation of the magnetization directions of two thin magnetic layers, where the output is determined by the high or low resistance value measured by the MTJ. In a GMR-based MTJ structure, however, only one operand of the XOR logic can be variable as the magnetization of the fixed layer is constant. This problem can be overcome by constructing a new read-only-port, where two free layers and one insulator layer are stacked.

The two free layers each have the size of a magnetization domain and are obtained from two different nanowires. Thus, the two operands, representing the magnetization direction in each free layer, can both be variables. Their values can be assigned through the MTJs of their own nanowire, and then shifted to the operating port to carry out the XOR operation.

The bitwise-XOR logic for the AddRoundKey operation implemented by two DW nanowires is shown in Figure 8. Let A and B be the 1-bit operands from the state byte and key byte, respectively. Eight identical DW-XORs are used for the bit-wise XOR operation between the state byte and the key byte. The following steps are involved in the execution of the operation $state \oplus key$.

- The binary operands, A and B , are loaded into two nanowires by enabling WR_1 and WR_2 , respectively;
- A and B are shifted from their access-ports to the read-only-port by enabling SHF_1 and SHF_2 , respectively;
- By enabling RD , the bitwise-XOR result can be obtained from the read-only-port through the GMR-effect.

The latency t_{ARK} in terms of the number of cycles consumed in the AddRoundKey stage can be calculated by:

$$t_{ARK} = (t_{read} + t_{xor} + t_{write}) \cdot \frac{128}{N_{xor}} \quad (7)$$

Fig. 9. MixColumns transformation by DW-LUT and DW-XOR

where $N_{xor} \in \{1, 2, 4, 8, 16, 32\}$.

Similar to the SubBytes transformation, using more DW-XOR units and SAs will increase the parallelism and reduce the latency for this stage. When $N_{xor} = 4 \times 8 = 32$, the maximum parallelism is reached and one complete column of M_s is processed simultaneously in the AddRoundKey stage.

E. MixColumns

The MixColumns transformation can be expressed as a multiplication of the state matrix M_s by the known matrix M_{mc} , as shown in Figure 1. The i -th column, $i \in \{0, 1, 2, 3\}$, after this transformation becomes

$$\begin{aligned} S'_{0,i} &= 2 \times S_{0,i} \oplus 3 \times S_{1,i} \oplus S_{2,i} \oplus S_{3,i} \\ S'_{1,i} &= S_{0,i} \oplus 2 \times S_{1,i} \oplus 3 \times S_{2,i} \oplus S_{3,i} \\ S'_{2,i} &= S_{0,i} \oplus S_{1,i} \oplus 2 \times S_{2,i} \oplus 3 \times S_{3,i} \\ S'_{3,i} &= 3 \times S_{0,i} \oplus S_{1,i} \oplus S_{2,i} \oplus 2 \times S_{3,i} \end{aligned}$$

where $S_{i,j}$ and $S'_{i,j}$ denote the values of the byte in the i -th row and j -th column of the state matrix before and after the MixColumns transformation, respectively.

The MixColumns transformation can be performed merely with DW-LUT and DW-XOR, as shown in Figure 9. The operations involved are multiplication by two (denoted by $xtime(2)$), multiplication by three (denoted by $xtime(3)$) and bit-wise XOR. Different from the normal integer multiplication, $xtime(2)$ is obtained by left shifting the operand by one bit, followed by bit-wise XORing it with $0x1B$ ¹ if the MSB of the operand is 1 before shifting; $xtime(3)$ is obtained by XORing the operand with its result of $xtime(2)$. The bit-wise XOR can be executed by the proposed DW-XOR. Although $xtime(2)$ can be implemented by in-memory shifting with additional DW-XOR, it is more efficient to be implemented by an 8-bit input, 8-bit output DW-LUT due to the associated branching operation that is dependent on the value of its MSB.

Unlike the SubBytes and AddRoundKey stages, where the transformation for each byte $S_{i,j}$ is independent of other bytes, each output byte of $S'_{i,j}$ in MixColumns is a weighted combination of four bytes in a column of M_s . Therefore it

¹ $0x1B$ corresponds to the irreducible polynomial $x^8 + x^4 + x^3 + x + 1$ in $GF(2^8)$ with the highest degree term discarded. This is because the term with the highest degree and the carry add modulo 2 to 0.

Algorithm 2: High Throughput/Energy Efficiency Domain-wall Nanowire based AES Mapping

Input: AES design area constraint A_m or/and power constraint P_m

- 1: $S_R \leftarrow \emptyset$
- 2: $\mathcal{R}_{SB} \leftarrow \{32 \times M_{nw}\}; \mathcal{R}_{MC} \leftarrow \{4 \times M_{lut}, 8 \times M_{4-16dec}\}$
- 3: **for** (N_{xor}, N_{lut}) combinations, $N_{xor} \in \{1, 2, 4, \dots, 32\}$ and $N_{lut} \in \{1, 2, 4\}$ **do**
- 4: $\mathcal{R}_{ARK} \leftarrow \{N_{xor} \times M_{xor}, 32 \times M_{sa}\}$
- 5: $\mathcal{R}_{SB} \leftarrow \{N_{lut} \times M_{lut}, 32 \times M_{sa}, 8 \times M_{4-16dec}\}$
- 6: calculate $\{t_{SB}, t_{ARK}, t_{SR}, t_{MC}\} \leftarrow f(\{\mathcal{R}_{SB}, \mathcal{R}_{ARK}, \mathcal{R}_{SR}, \mathcal{R}_{MC}\})$
- 7: $t'_{cycle} \leftarrow \text{Sum}\{t_{SB}, t_{ARK}, t_{SR}, t_{MC}\}$
- 8: $S_R \leftarrow \mathcal{R}_{SB} \cup \mathcal{R}_{ARK} \cup \mathcal{R}_{SR} \cup \mathcal{R}_{MC}$
- 9: calculate area $A \leftarrow g(S_R)$, power $P \leftarrow h(S_R)$, and throughput $T' \leftarrow j(S_R)$
- 10: **if** $A \leq A_m$ **and** $P \leq P_m$ **and** $t'_{cycle} \leq t_{cycle}$ **then**
- 11: $S_R \leftarrow S'_R$
- 12: $t_{cycle} \leftarrow t'_{cycle}$
- 13: **end if**
- 14: **end for**

Output: DW nanowire based AES architecture resource utilization S_R

cannot be further parallelized by having more DW-LUT units and DW-XOR. The number of cycles t_{MC} consumed in the *MixColumns* stage can be calculated as follows:

$$t_{MC} = (t_{read} + t_{LUT} + 3 \cdot t_{xor} + t_{write}) \cdot 4 \quad (8)$$

Due to the multiple-cycle operation of DW-XOR, and higher number of DW-XORs in the critical path, *MixColumns* is the most time consuming stage among the four stages.

F. KeyExpansion

AES in Figure 1 assumes a 128-bit key but the input data block needs to undergo ten rounds of AddRoundKey operation. To achieve the objective of confusion, the KeyExpansion function needs to generate a different round key for every iteration of AddRoundKey operation. For higher level of security, 192- and 256-bit keys with 12 rounds and 14 rounds of mixing of the same 128-bit input block are applied.

The KeyExpansion operation itself is composed of circular rotation, byte substitution, XOR operation with the round constant and word-level XOR [26], which can all be decomposed into the four previously introduced basic operations. In other words, the KeyExpansion function for all three key lengths of AES can be mapped to the previously described operations of DW nanowire devices. Algorithm 2 provides the hardware mapping of overall AES operations to DW devices for given design constraints.

G. DW-AND Operation for GCM Architecture Extension

Among different modes of operation for symmetric key cryptographic block ciphers, Galois Counter Mode (GCM) has been widely adopted due to its efficiency and performance. As an efficient authenticated encryption algorithm, AES-GCM [27] can fully exploit the parallel and pipelined processing of the original AES in Figure 1 to provide both data authenticity and confidentiality. Besides all the operations required in Figure 1, a GCM hash function is needed to generate the authentication tag. The Galois Field (GF) multiplication of GCM hash function can be obtained by ANDing the authenticated data A and the ciphertext data C with a hash key H , where A , C and H are all 128-bit data. This 128×128 AND-array can

be implemented using a DW memory based AND operation. Figure 10 shows the detailed implementation of a 1-bit AND operation on the DW nanowire. To perform an AND operation, we need to pre-write a 0 into a domain of the DW nanowire first, and then write a bit of A or C into the next domain. The corresponding bit in H is then applied on the shift transistor. If the hash key bit is 0, the DW nanowire will not shift and the pre-write 0 will be read out. If the data in A or C is 0, the readout data will always be 0 irrespective of whether there is any DW movement in the nanowire. Altogether 128 pairs of 1-bit AND DW nanowire are needed. The GF sum of the partial product bits can be easily realized by the DW-XOR structure as in the AddRoundKey stage shown in Fig. 8. Hence the GCM operation can be fully integrated into the DW nanowire based AES.

H. Security Analysis

Fault analysis and power analysis based side-channel attacks are the major threats in CMOS AES implementations [28], [29]. Our proposed architecture is inherently resilient against these two attacks owing to the use of spintronic devices and in-memory computing architecture. For the fault analysis attack, an external magnetic field is required to inject the fault because the logic and storage are implemented by magnetic materials. Due to the size of the nanomagnets, it is impossible to localize the applied field to influence only a single MTJ or a DW nanowire within a densely integrated magnetic memory array to precisely inject a fault into a target cell. As opposed to charge-based devices, spin-based devices are normally off, they have zero leakage and ultra-low switching power. Moreover, as all the AES logic are implemented inside the memory, it is difficult to correlate the indistinguishable power dissipation of each logic block measured in different time slots of encryption.

V. PIPELINED AES BY DOMAIN-WALL NANOWIRE

A. Pipelined DW-AES

Figure 11(a) shows the DW-AES implementation without pipelining. There are four stages, namely *SubBytes*, *AddRoundKey*, *ShiftRows* and *MixColumns*, and each stage has

Fig. 10. AND logic realized by domain-wall nanowire for AES-GCM

different latency. Only one of the four stages is active at one time while the other three stages are idle. The idle stages will consume leakage power even if they do not process any data. Pipelining can be introduced to exploit the idle resources and increase their utilization rate in order to achieve higher energy efficiency and throughput. For DW-AES architecture, pipelining is not as straightforward as CMOS ASIC implementation since each DW logic stage requires different numbers of cycles. Based on the latencies of six, two, eleven and five cycles for *SubBytes*, *AddRoundKey*, *ShiftRows* and *MixColumns*, respectively, the throughput of the pipeline is limited by the most time consuming *ShiftRows* stage. The *AddRoundKey* result has to wait for nine cycles before it can accept new input data from the *SubBytes* stage, and transfer its result to the next *ShiftRows* stage. Similarly, the *MixColumns* stage also has to wait until *ShiftRows* has completed its computation in order to feed the result to it.

Instead of using timers to control the data flow for pipelining, first-in-first out (FIFO) buffers adapted to different cycle delays of each stage can be designed by exploiting the tandem movement of domain walls in nanowire. Figure 11(b) shows the timing diagram of pipelined DW-AES with the use of DW-FIFOs. The DW-FIFOs between stages are illustrated in Figure 11(c), which are essentially nanowires of different lengths. By changing the length of nanowire and shifting one domain of the nanowire per cycle, any number of cycles can be achieved. As shown in Figure 11(b), the latency of every stage in a pipelined DW-AES can be equalized with the insertion of an appropriate length of DW-FIFO between stages. Thus the data can be fed in regular interval without requiring sophisticated

TABLE I
NOTATIONS FOR DOMAIN-WALL DEVICE BASED DW-AES
IMPLEMENTATION

Symbols	Descriptions
M_s	4×4 state matrix for the input of 128-bit AES
$S_{i,j}$	State byte, entry in the i -th row and j -th column of M_s
$K_{i,j}$	key bytes correspond to $S_{i,j}$
A_i	the i -th DW array for physical implementation of M_s
N_r	number of iterative rounds in AES algorithm
M_{mc}	the MixColumns transformation matrix
T	throughput of AES cipher
N_{LUT}	number of look-up tables
N_{xor}	number of xor units
t	latency of DW-AES measured in number of cycles
$\mathcal{S}_R$	set of resources utilized by DW AES architecture
$\mathcal{R}$	set of resources for each module
M	memory based logic units in the resource sets

multirate signal processing technique or multi-clock domain synchronization.

For 128-bit AES, each input data needs to undergo the four stages of operations for ten times. For a four-stage pipeline, the fifth data chunk can only begin after all ten iterations have been completed. This problem can be solved by generating multiple ciphers in series to maintain the same input data rate without the need for extra buffering of input data.

The algorithm for the mapping of DW-AES to a high-speed energy-efficient non-pipelined architecture is described in Algorithm 2. The symbols used are listed in Table I. Specifically, it enumerates different amount of resources (e.g., N_{xor} and N_{LUT}) and allocates the resources to each stage (e.g., $\mathcal{R}_{ARK}$ is a set of resources for *AddRoundKey* stage). Then the delay, power and area of each stage are evaluated according to each operation detailed in Section IV. Lastly, the resource configuration with the best target performance (power, area or throughput) will be selected as the final solution $\mathcal{S}_R$ for the hardware performance specified in the design stage. In general, the maximum throughput of DW-AES design without pipelining can be obtained by minimizing the total latency as follows:

$$\begin{aligned}
 t_{MinSum} = & \underset{\mathcal{S}_R}{\text{minimize}} \quad \text{Sum}(t_{SB}, t_{ARK}, t_{SR}, t_{MC}) \\
 \text{subject to} \quad & P(\mathcal{S}_R) \leq P_m \\
 & A(\mathcal{S}_R) \leq A_m,
 \end{aligned} \tag{9}$$

where t_{SB} , t_{ARK} , t_{SR} and t_{MC} are the numbers of cycles required for the execution of *SubBytes*, *AddRoundKey*, *ShiftRows* and *MixColumns* stages determined by the allocated resources $\mathcal{S}_R$. P_m and A_m are the maximum allowable power and area specified by the design constraints. Consequently, the throughput of non-pipelined DW-AES, $T \propto \frac{1}{t_{MinSum}}$.

For the pipelined DW-AES architecture, the throughput is limited by the critical stage that has the longest latency. The maximum throughput of pipelined DW-AES design is obtained by minimizing the latency of the most time-critical stage as follows

$$\begin{aligned}
 t_{input} = & \underset{\mathcal{S}_R}{\text{minimize}} \quad \text{Max}(t_{SB}, t_{ARK}, t_{SR}, t_{MC}) \\
 \text{subject to} \quad & P(\mathcal{S}_R) \leq P_m \\
 & A(\mathcal{S}_R) \leq A_m,
 \end{aligned} \tag{10}$$

The corresponding throughput of DW-AES is $T_{pipeline} \propto$

Fig. 11. (a) DW-AES without pipelining; (b) Pipelined DW-AES by DW-FIFO insertion; (c) Adjusting the latency of DW-FIFOs to balance the pipelined stages

Algorithm 3: Pipelined DW Nanowire based AES Mapping

Input: AES design area constraint A_m or/and power constraint P_m

- 1: $S_R \leftarrow \emptyset$
- 2: $\mathcal{R}_{SR} \leftarrow \{32 \times M_{nw}\}; \mathcal{R}_{MC} \leftarrow \{4 \times M_{LUT}, 8 \times M_{4-16dec}\}$
- 3: **for** (N_{xor}, N_{LUT}) combinations, $N_{xor} \in \{1, 2, 4, \dots, 32\}$ and $N_{LUT} \in \{1, 2, 4\}$ **do**
- 4: $\mathcal{R}_{ARK} \leftarrow \{N_{xor} \times M_{xor}, 32 \times M_{sa}\}$
- 5: $\mathcal{R}_{SB} \leftarrow \{N_{LUT} \times M_{LUT}, 32 \times M_{sa}, 8 \times M_{4-16dec}\}$
- 6: calculate $\{t_{SB}, t_{ARK}, t_{SR}, t_{MC}\} \leftarrow f(\{\mathcal{R}_{SB}, \mathcal{R}_{ARK}, \mathcal{R}_{SR}, \mathcal{R}_{MC}\})$
- 7: $\underset{N \in P}{\text{minimize}} \text{Max}\{\frac{t_{SB}}{N_{SB}}, \frac{t_{ARK}}{N_{ARK}}, \frac{t_{SR}}{N_{SR}}, \frac{t_{MC}}{N_{MC}}\}$
- 8: $S'_R \leftarrow \{N_{SB} \times \mathcal{R}_{SB}\} \cup \{N_{ARK} \times \mathcal{R}_{ARK}\} \cup \{N_{SR} \times \mathcal{R}_{SR}\} \cup \{N_{MC} \times \mathcal{R}_{MC}\}$
- 9: calculate area $A \leftarrow g(S'_R)$, power $P \leftarrow h(S'_R)$ and throughput $T' \leftarrow j(S'_R)$
- 10: **if** $A \leq A_m$ **and** $P \leq P_m$ **and** $T' \geq T$ **then**
- 11: $S_R \leftarrow S'_R$
- 12: **end if**
- 13: **end for**

Output: Pipelined DW-AES architecture resource utilization S_R

$\frac{1}{t_{input}}$. Ideally, if the four stages have the same latency, the throughput of the pipelined DW-AES will be quadrupled. The gain in throughput by pipelining diminishes with increasing latency skew of the critical stage. Taking into account the resource constraint, and limited power and area budgets, the computational resources will need to be relocated from the non-critical stages to the critical stage to improve the latency of the latter. Design space exploration can be performed to find the best pipelined DW-AES configuration with the highest achievable throughput.

B. Multi-issue Pipelined DW-AES

Due to the large disparity in computational complexity of different pipelined stages, resource reallocation may not be able to adequately reduce the latency skew among all stages. Hence, we propose to adopt the multiple issue technique used

in super-scalar processor for the pipelined DW-AES. The DW-AES with pipeline/multi-issue mapping algorithm is described in Algorithm 3 using the symbols defined in Table I. To overcome the timing bottleneck, additional processing units are introduced to share the workload of any critical stages after the resource reallocation, as shown in Figure 12(b). In Figure 12(b), two additional units are added to the bottleneck MixColumns stage. Its timing diagram is shown in Figure 12(a), which illustrates the throughput improvement of the overall pipelined architecture. In Figure 12(a), it is observed that the MixColumns stage is twice slower than other stages. Before the application of multiple issue technique, the latency of MixColumns is three time higher than other stages, and the data can only be fed at a rate that is limited by its latency. By adding two more MixColumns units, the MixColumns stage is able to operate at the same speed as other stages, thus the data rate is tripled as the latency of all stages are now equalized. In general, the period of data input clock t_{input} shown in Figure 12(a) can be obtained by

$$t_{input} = t_{MinMax} =$$

$$\underset{S_R}{\text{minimize}} \text{Max}(\frac{t_{SB}}{N_{SB}}, \frac{t_{ARK}}{N_{ARK}}, \frac{t_{SR}}{N_{SR}}, \frac{t_{MC}}{N_{MC}}) \quad (11)$$

$$\text{subject to} \quad P(S_R) \leq P_m$$

$$A(S_R) \leq A_m,$$

where N_{SB} , N_{ARK} , N_{SR} and N_{MC} are the number of processing units of *SubBytes*, *AddRoundKey*, *ShiftRows* and *MixColumns* stages, respectively. The corresponding throughput for the multi-issue pipelined DW-AES is $T_{multiissue} \propto \frac{1}{t_{input}}$.

Compared to the pipelined DW-AES, the multi-issue pipelined DW-AES has a higher throughput. In addition, due to the higher utilization rate, less resources are idle most of the time. Consequently, the ratio of leakage power to total power consumption is lowered, which further improves the energy efficiency of the multi-issue pipelined DW-AES at the expense of area overhead due to the additional resources.

VI. SIMULATION RESULTS

A. Settings

To evaluate the proposed DW-AES cipher, the following design platform has been set up. First, at device level, the transient simulation of the read and write operations of MTJ are performed within NVM-SPICE [30] to obtain the accurate operation energy and timing for DW nanowire. The shift-operation energy is modeled as Joule heating on the nanowire when shift-current is applied. The shift-current density and shift-velocity relationships are based on [25]. The area of a DW nanowire is calculated by its dimension parameters. Specifically, 1V V_{DD} is used and the technology node of 32nm is assumed with the width of 32nm and the length of 64nm per domain, and a thickness of 2.2nm for one DW nanowire; the permittivity of MgO layer is $8.8\epsilon_0$ [31], which leads to a parasitic capacitance of $7.25e-17$ F for the MTJ; the resistances R_{off} and R_{on} are set to 2600Ω and 1000Ω , respectively, the write current to $100\mu A$ and the current density to $6 \times 10^8 A/cm^2$ for the shift operation. At circuit level, the memory modeling tool CACTI [32] is modified to perform

Fig. 12. (a) Timing diagram and (b) block diagram of pipelined DW-AES with multi-issue

Fig. 13. Timing diagram of DWL-XOR with SPICE-level simulation for each operation

accurate power and area calculation for DW nanowire memory peripheral circuits, such as decoders and SAs. The modified tool is called DW-CACTI. With the device level performance data, DW-XOR and DW-LUT can be evaluated at circuit level. The additional sequential controller of DW-AES is described by Verilog HDL, and synthesized to obtain the area and power information. Finally, at the system level, an AES behavioral simulator is developed to explore the design space for optimal trade-off among power, area and speed. As the in-memory architecture mainly serves as memory, and the in-memory logic only executes upon request, we assume that the in-memory AES logic is active 10% of the time and normal memory accesses account for 90% of the total usage.

B. Domain-Wall Logic Performance

Table II shows the energy cost and speed of the basic operations of DW nanowire. The sub-pJ results for the MTJ read and write operations obtained from NVM-SPICE [22], [23] agree with the reported measurement data in [33]. The current induced shift result also agrees with the reported measurement data in [21]. The transient analysis of the DW nanowire XOR structure has been performed in SPICE. The timing diagrams of the control signals are shown in Figure 13.

The current density of $7e10A/m^2$ is utilized for the magnetization switching. The azimuthal angles θ of magnetization orientation of the nanowire for operand *A* are all initialized at 0, and those for operand *B* are all set at π . Only two bits per nanowire is assumed for both nanowires. The output-port is implemented as a developed MCM device, with the internal state variables θ and ϕ for the magnetization orientations of the upper and bottom layers, respectively. During the *loadA* and *loadB* cycles, 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, respectively. These values are consistent with those reported for the devices in

TABLE II
PERFORMANCES OF THE BASIC OPERATIONS OF DW NANOWIRE

Operation	read	write	shift	XOR	LUT
speed (cycles)	1	1	1	5	3
energy (pJ)	0.06	0.1	0.03	0.26	0.28

[33]. In the *shift* cycles, triggered by the *SHF*-control signal, the dynamics of θ and ϕ of both the upper and bottom layers are updated immediately. In the *operation* cycle, a subtle sensing current is applied to provoke the GMR-effect. Subtle magnetization disturbance is also observed in both layers of the MCM device, which validates the read operation. The change in θ values from their initial values in the *operation* cycle also validates the successful shifting of domain wall.

C. AES Performance Comparison

The proposed DW-AES cipher is compared with both CMOS-based ASIC design [4],[6] and hybrid CMOS/ReRAM (CMOL) design [7]. In ASIC implementation, the performance data is extracted from the reported results in [4],[6] with the following technology scaling: $Area \sim 1/S^2$, $Delay \sim 1/S^2$, $Energy \sim 1/S$, where $S = L/32nm$ and leakage scaling factor from [34]. The stacked CMOL AES performance is projected from [7]. In particular, the lower layer CMOS gates are evaluated with the aforementioned technology scaling together with the interconnection improvement ratio by the upper non-volatile layer. C-code based software implementation that runs on a general purpose processor (GPP) is also compared. The evaluation of the software implementation of AES is done in two steps. Firstly, the AES binary compiled by the C-code obtained from [35] is input to gem5 [36] simulator to generate the run-time utilization rate of core components. Then, the generated statistics are used by McPAT [37] to produce the core power and area model. Table III compares the different implementations of AES cipher, and the results are discussed as follows.

1) *Power*: As expected, the DW-AES cipher has the smallest leakage power due to the use of non-volatile DW nanowire devices. The small leakage power is mainly introduced by its CMOS peripheral circuits, i.e., decoders, SAs, as well as simple controllers. Specifically, DW-AES cipher reduces the leakage power of CMOS ASIC and memristive CMOL designs by 96% and 98%, respectively. The leakage power can be further reduced if the decoders and SAs of the memory are reused by the DW-AES ciphers. All three types of DW-AES consume comparable energy for the encryption of one 128-bit input block. The slight differences are ascribed to the DW-FIFO and different CMOS-based cycle controllers. Despite consuming slightly higher leakage power than the baseline

TABLE III
COMPARISON OF DIFFERENT AES IMPLEMENTATIONS ON 128-BIT ENCRYPTION PERFORMANCE

Implementation	leakage (μW)	energy /128-bit (nJ)	area (μm^2)	cycles
C code [35] on GPP [†]	1.3e6	460	2.5e+6	2309
CMOS ASIC [4]	400	6.6	4400	336
memristive CMOL [7]	624	10.3	320	470
CMOS Pipelined [6]	267-16800 [‡]	0.9	75000	20
Baseline DW-AES	14.6	2.4	78	1022
Pipelined DW-AES	15.3	2.3	83	2652
Multi-issue DW-AES	21.4	2.7	155	1320

[†] general purpose processor

[‡] $267\mu W @ V_{DD}=320$ mV, and $16.8mW @ V_{DD}=1.1V$

Fig. 14. The breakdown of (a) area (b) latency (c) dynamic power (d) leakage power consumed by every module of DW-AES, pipelined DW-AES and multi-issued DW-AES

DW-AES, pipelined DW-AES and multi-issue DW-AES encrypt four input blocks each time, which result in a higher overall energy efficiency. This is because the computational resources of pipelined DW-AES and multi-issue DW-AES are active most of the time. The energy is expended for useful functions than wasted in idleness. The breakdowns of dynamic and leakage power consumptions for each module of all DW based implementations are shown in the bar charts of Figure 14(c) and 14(d), respectively. The *MixColumns* module that involves more xor and LUT operations consume the majority of the total dynamic power. On the other hand, leakage power is dominated by the *MixColumns* and *SubBytes* modules due to their predominant volatile components such as decoders for DW memory array and SAs.

2) *Area*: Benefiting from the high density of DW nanowire devices, DW-AES is highly area-efficient. In particular, substantial areas have been saved by employing very compact DW-LUTs in the two most resource consuming *SubBytes* and *MixColumns* modules.

As mentioned earlier, the areas reported in [4], [6] and [7] are technologically scaled to 32 nm technology by a factor of $1/S^2$. The areas of our proposed architectures are calculated based on the number of CMOS transistors, and the sizes of MTJ, DW nanowire and memory array using the DW-CACTI tool developed from CACTI [32]. As the in-memory architecture is composed of CMOS and DW nanowire circuits, the area consumed in each part is calculated separately. For example, the DW-LUT in Figure 6 requires a 256-byte DW memory and two 4-16 decoders. With each domain occupying an area of $32nm \times 64nm$, the areas of the DW memory and the two decoders are calculated to be $4.2\mu m^2$ and $0.95\mu m^2$, respectively. Because the DW nanowire and CMOS circuits are implemented in different layers, the much smaller CMOS circuit area is fully subsumed under DW nanowire stack. Hence the *SubBytes* stage is dominated by the larger nanowire circuit area of $4.2\mu m^2$. The areas of *ShiftRows*,

AddRoundKey and *MixColumns* are calculated in the same way to be $0.04\mu m^2$, $3.3\mu m^2$ and $0.3\mu m^2$, respectively. As the total area of each encryption cipher is only $7.8\mu m^2$, 10 sets of encryption ciphers are implemented in our design. Pipelined DW-AES and multi-issue DW-AES occupy larger areas than the baseline DW-AES due to the additional register arrays for pipeline and hardware resources like 3 *MixColumns* for multi-issue.

Overall, the DW-AES cipher saves 98% and 76% of areas over the CMOS ASIC and memristive CMOL designs, respectively. Compared to DW-AES, the pipelined DW-AES incurs a small area overhead of 6.6% due mainly to the DW-FIFO used for stage balancing and additional state matrices. The area of multi-issue DW-AES is almost twice that of DW-AES. This is attributable to the additional processing units. Nevertheless, it still exhibits a noteworthy 52 % area reduction compared to CMOL design. The breakdown of areas consumed by different modules of all DW based implementations is shown in Figure 14(a). Due to the use of DW-LUT, *SubBytes* and *MixColumns* consume almost all of the area.

3) *Latency*: The premium for the higher area and energy efficiency of DW-AES is an increase in the number of cycles required for its computation. Multiple-cycle operations of DW-XOR and its DW-LUT are inevitable as these functions need to be preceded by the shift-operation in order to align the target cell with the MTJ. The breakdown of latencies required by different modules for all DW based implementations is shown in Figure 14(b). Additional cycles are added by DW-FIFO to balance the number of cycles of all pipelined stages for the pipelined DW-AES. The latency of multi-issue DW-AES is only slightly larger than that of DW-AES because stage balancing is augmented by adding processing units, with only a small number of cycles added by DW-FIFO. Although the latency from the input of plaintext to the output of cipher text is longer in pipelined DW-AES and multi-issue DW-AES, they are able to process four times more data than the baseline DW-AES. In other words, the extended latency of these pipelined designs is compensated by a higher processing rate. After all, the overall speed of the cipher is determined by how fast a stream of plaintext data can be encrypted continuously instead of how long it takes to encrypt only one chunk of 128-bit data.

TABLE IV
SYSTEM CONFIGURATIONS

AES computing platforms configurations and Date Processing Rate (DPR) under $2mm^2$ area design budget				
platforms	# of AES ciphers	clock-rate	critical stage latency (for 128 bits)	DPR (GB/s)
C code [35] on GPP	1 core	2GHz	2309	0.016
CMOS ASIC [4]	454	30MHz	84	2.59
Pipelined ASIC [6]	26		5	2.5
memristive CMOL [7]	6250		470	6.38
DW-AES	25640		1022	12
DW-AES pipelined	24096		663	17.4
DW-AES multi-issue	12902		220	28
memory configurations				
memory capacity	1GB			
bus width	128 bits			
bandwidth	6.4 GB/s			
I/O energy overhead	3.7nJ per access

Fig. 15. Comparison of data processing rate (DPR), power and energy efficiency of encryption among different in-memory AES computing platforms

D. Data Processing Rate and Energy Efficiency Comparison

In what follows, the proposed in-memory DW-AES is compared with other implementations at the system level. For each AES computing platform, the number of AES units is maximized to encrypt the input data stream concurrently subject to a fixed area constraint. Except the proposed in-memory DW-AES, all platforms will incur some I/O energy overhead for data access. Given a 2mm^2 area design budget, the system configurations for different platforms are summarized in Table IV. For more compact implementation such as [6], more AES ciphers can be housed within the 2mm^2 area budget. The memory I/O energy overhead is obtained from CACTI. The processor operates at 2GHz for software implementation, while all other AES accelerators run at a typical clock-rate of 30MHz, which achieves the optimal energy efficiency according to [4], [6]. Operating the accelerators at faster clock-rate will adversely affect the systems reliability due to high peak power for densely distributed and concurrent execution. Due to the number of concurrent encryptions within a small footprint, 30MHz is a reasonable clock rate according to Figure 15(b) for all custom circuit implementations to avoid overheating. Given a uniform operating frequency, the number of cycles for the most time-critical stage is different among all implementations, which is indicated in Table IV for a 128-bit cipher. A figure of merit, called the data processing rate (DPR), is defined to measure the rate of encrypted data within a given area budget for different custom implementations. DPR, expressed in bytes per second, is defined as

$$\text{DPR} = \frac{\# \text{ciphers} \times f \times \# \text{bytes per cipher}}{\text{latency of critical stage}}, \quad (12)$$

where $\# \text{ciphers}$ is the number of ciphers that can be implemented in a given area, f is the clock frequency in Hz, $\# \text{bytes per cipher}$ is the number of bytes in each cipher and the latency of critical stage is the number of clock cycles consumed in the longest stage of AES implementation.

Figure 15 compares the DPR, power and energy efficiency of different AES computing platforms. As expected, all AES hardware implementations have several orders of magnitude improvement in DPR and energy efficiency compared to the software implementation on GPP. Among all the hardware implementations, the proposed DW-AES has a DPR of 12 GB/s, which is $4.6\times$ higher than that of the CMOS ASIC

based platform with a similar power consumption, and $1.88\times$ higher than that of the memristive CMOL based platform with 38% power reduction. Due to the smaller area per cipher of DW-AES, more ciphers can operate in parallel under the same area budget, leading to its higher DPR. The multi-issue pipelined DW-AES has the highest DPR of 28 GB/s, which is $11\times$ higher than its CMOS ASIC counterpart with $2.2\times$ power penalty. This further improvement in DPR is achieved by reducing the latency of the critical stage. Due to the in-memory encryption and non-volatility, the proposed DW-AES computing platforms have the best energy efficiency of 22pJ/bit, which is $5\times$, $5.2\times$, $3\times$ times higher than that of CMOS ASIC, pipelined CMOS ASIC and memristive CMOL based platforms, respectively.

For higher usage of in-memory AES logic, the dynamic power has larger impact on the energy efficiency; while for lower usage of in-memory logic, the leakage power has greater impact. As the DW-AES implementations have comparable energy consumption but much lower leakage power, as shown in Table III, the energy efficiency will increase when the usage rate decreases. For high usage, the advantage of energy efficiency of DW-AES ciphers over ASIC implementations will be somewhat reduced but it remains competitively better owing to its efficient in-memory communication and lower energy consumption per encryption.

VII. CONCLUSION

A block level in-memory architecture for AES encryption has been proposed. All the logic operations required for the computation of AES cipher are realized within the memory array using the DW nanowires that feature ultra-low leakage power dissipation. The predominant XOR gates in CMOS-based AES are replaced by DW-XOR gates, the ShiftRows transformation by DW shifter and the S-box function by DW-LUT. The spintronic devices used are similar to the devices used for the non-volatile storage elements, which make the integration of the complete in-memory AES computing architecture highly uniform and compact. The throughput of the proposed DW-AES can be boosted by pipelined and multi-issue techniques with the insertion of SW-FIFO and additional DW nanowires computing units for stage balancing. Our experiment results show that the proposed pipelined DW-AES exhibits the best energy efficiency of 22 pJ/bit among its

rivals, with $5.2\times$ and $3\times$ improvements over CMOS ASIC and memristive CMOL based AES implementations, respectively.

REFERENCES

- [1] D. B. Strukov *et al.*, "The missing memristor found," *Nature*, vol. 453, no. 7191, pp. 80–83, May 2008.
- [2] T. Kawahara *et al.*, "2Mb spin-transfer torque RAM (SPRAM) with bit-by-bit bidirectional current write and parallelizing-direction current read," in *Proc. 2007 IEEE Int. Solid-State Circuits Conf., San Francisco, CA*, Feb. 2007, pp. 480–617.
- [3] X. Wang *et al.*, "Spin torque random access memory down to 22 nm technology," *IEEE Trans. Magn.*, vol. 44, no. 11, pp. 2479–2482, Nov. 2008.
- [4] S. Mathew *et al.*, "340mV-1.1V, 289Gbps/W, 2090-gate NanoAES hardware accelerator with area-optimized encrypt/decrypt $GF(2^4)^2$ polynomials in 22nm tri-gate CMOS," in *Proc. 2014 Symp. VLSI Circuits, Honolulu, HI*, Jun. 2014, pp. 1–2.
- [5] M. Mozaffari-Kermani and A. Reyhani-Masoleh, "Efficient and high-performance parallel hardware architectures for the AES-GCM," *IEEE Trans. Comput.*, vol. 61, no. 8, pp. 1165–1178, Aug. 2012.
- [6] S. Mathew *et al.*, "53 Gbps native $GF(2^4)^2$ composite-field AES-encrypt/decrypt accelerator for content-protection in 45 nm high-performance microprocessors," *IEEE J. Solid-State Circuits*, vol. 46, no. 4, pp. 767–776, Apr. 2011.
- [7] Z. Abid *et al.*, "Efficient CMOL gate designs for cryptography applications," *IEEE Trans. Nanotechnol.*, vol. 8, no. 3, pp. 315–321, May 2009.
- [8] S. Matsunaga *et al.*, "MTJ-based nonvolatile logic-in-memory circuit, future prospects and issues," in *Proc. 2009 Design, Autom. Test Eur., Nice, France*, Apr. 2009, pp. 433–435.
- [9] H. Kimura *et al.*, "Complementary ferroelectric-capacitor logic for low-power logic-in-memory VLSI," *IEEE J. Solid-State Circuits*, vol. 39, no. 6, pp. 919–926, Jun. 2004.
- [10] H.-P. Trinh *et al.*, "Domain wall motion based magnetic adder," *Electron. Lett.*, vol. 48, no. 17, pp. 1049–1051, Aug. 2012.
- [11] S. Matsunaga *et al.*, "Fabrication of a nonvolatile full adder based on logic-in-memory architecture Using magnetic tunnel junctions," *Appl. Phys. Express*, vol. 1, no. 9, p. 1301, Sep. 2008.
- [12] W. Zhao *et al.*, "New non-volatile logic based on spin-MTJ," *physica status solidi (a)*, vol. 205, no. 6, pp. 1373–1377, Jun. 2008.
- [13] N. Sakimura *et al.*, "Nonvolatile magnetic flip-flop for standby-power-free SoCs," *IEEE J. Solid-State Circuits*, vol. 44, no. 8, pp. 2244–2250, Aug. 2009.
- [14] Y. Wang *et al.*, "Energy efficient in-memory AES encryption based on nonvolatile domain-wall nanowire," in *Proc. 2014 Design, Autom. Test Eur., Grenoble, France*, Mar. 2014, pp. 1–4.
- [15] S. S. Parkin, M. Hayashi, and L. Thomas, "Magnetic domain-wall racetrack memory," *Science*, vol. 320, no. 5873, pp. 190–194, Apr. 2008.
- [16] R. Venkatesan *et al.*, "TapeCache: a high density, energy efficient cache based on domain wall memory," in *Proc. 2012 ACM/IEEE Int. Symp. Low Power Electron. and Design, Redondo Beach, CA*, Aug. 2012, pp. 185–190.
- [17] L. Thomas *et al.*, "Racetrack Memory: A high-performance, low-cost, non-volatile memory based on magnetic domain walls," in *Proc. 2011 IEEE Int. Electron Devices Meeting, Washington, DC*, Dec. 2011, pp. 24.2.1–24.2.4.
- [18] R. Usselman. (2002) Advanced Encryption Standard / Rijndael IP Core. [Online]. Available: http://opencores.org/project,aes_core
- [19] L. Ni, Y. Wang, H. Yu, W. Yang, C. Weng, and J. Zhao, "An energy-efficient matrix multiplication accelerator by distributed in-memory computing on binary rram crossbar," in *IEEE ASP-DAC*, 2016.
- [20] M. Stiles and J. Miltat, "Spin-transfer torque and dynamics," in *Spin dynamics in confined magnetic structures III*, B. Hillebrands and A. Thiaville, Eds. Berlin, Germany: Springer, 2006, pp. 225–308.
- [21] D. Chiba *et al.*, "Control of multiple magnetic domain walls by current in a Co/Ni nano-wire," *Appl. Phys. Express*, vol. 3, no. 7, p. 073004, Jul. 2010.
- [22] W. Fei *et al.*, "Design exploration of hybrid CMOS and memristor circuit by new modified nodal analysis," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 20, no. 6, pp. 1012–1025, Jun. 2012.
- [23] Y. Shang, W. Fei, and H. Yu, "Analysis and modeling of internal state variables for dynamic effects of nonvolatile memory devices," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 59, no. 9, pp. 1906–1918, Sep. 2012.
- [24] Y. Nogami, K. Nekado, T. Toyota, N. Hongo, and Y. Morikawa, "Mixed bases for efficient inversion in $F((2^2)^2)$ and conversion matrices of subbytes of aes," in *Cryptographic Hardware and Embedded Systems, CHES 2010*. Springer, 2010, pp. 234–247.
- [25] C. Augustine *et al.*, "Numerical analysis of domain wall propagation for dense memory arrays," in *Proc. 2011 IEEE Int. Electron Devices Meeting, Washington, DC*, Dec 2011, pp. 17.6.1–17.6.4.
- [26] J. Daemen and V. Rijmen, *The design of Rijndael: AES-the advanced encryption standard*. Berlin, Germany: Springer, 2002.
- [27] A. Satoh, T. Sugawara, and T. Aoki, "High-performance hardware architectures for galois counter mode," *Computers, IEEE Transactions on*, vol. 58, no. 7, pp. 917–930, 2009.
- [28] R. Karri, K. Wu, P. Mishra, and Y. Kim, "Fault-based side-channel cryptanalysis tolerant rijndael symmetric block cipher architecture," in *Defect and Fault Tolerance in VLSI Systems, 2001. Proceedings. 2001 IEEE International Symposium on*. IEEE, 2001, pp. 427–435.
- [29] G. Bertoni, L. Breveglieri, I. Koren, P. Maistri, and V. Piuri, "Error analysis and detection procedures for a hardware implementation of the advanced encryption standard," *Computers, IEEE Transactions on*, vol. 52, no. 4, pp. 492–505, 2003.
- [30] H. Yu and Y. Wang, "Nonvolatile state identification and NVM SPICE," in *Design Exploration of Emerging Nano-scale Non-volatile Memory*. New York, NY: Springer, 2014, pp. 45–83.
- [31] P. Padhan *et al.*, "Frequency-dependent magnetoresistance and magnetocapacitance properties of magnetic tunnel junctions with MgO tunnel barrier," *Appl. Phys. Lett.*, vol. 90, no. 14, pp. –, 2007.
- [32] S. Wilton and N. Jouppi, "CACTI: an enhanced cache access and cycle time model," *IEEE J. Solid-State Circuits*, vol. 31, no. 5, pp. 677–688, May 1996.
- [33] H. Zhao *et al.*, "Sub-200 ps spin transfer torque switching in in-plane magnetic tunnel junctions with interface perpendicular anisotropy," *J. Phys. D: Appl. Phys.*, vol. 45, no. 2, p. 025001, Dec. 2012.
- [34] Z. Abbas and M. Olivieri, "Impact of technology scaling on leakage power in nano-scale bulk CMOS digital standard cells," *Microelectr. J.*, vol. 45, no. 2, pp. 179 – 195, Feb. 2014.
- [35] K. Malbrain. (2009) Byte-oriented-aes: A public domain byte-oriented implementation of AES in C. [Online]. Available: <https://code.google.com/p/byte-oriented-aes/>
- [36] N. Binkert *et al.*, "The gem5 simulator," *ACM SIGARCH Comput. Architecture News*, vol. 39, no. 2, pp. 1–7, May 2011.
- [37] S. Li *et al.*, "McPAT: an integrated power, area, and timing modeling framework for multicore and manycore architectures," in *Proc. 2009 IEEE/ACM Int. Symp. Microarchitecture, New York, NY*, Dec. 2009, pp. 469–480.

Paper ID: T-IFS-05663-2015

Paper Title: *DW-AES: A Domain-wall Nanowire Based AES for High Throughput and Energy-efficient Data Encryption in Non-volatile Memory*

Authors: Yuhao Wang, Leibin Ni, Chip-Hong Chang, and Hao Yu

Response to the recommendations of Associate Editor

We would like to thank the anonymous reviewers for their valuable comments and constructive suggestions. We are glad to inform the Editor and Associate Editor that all the concerns raised by the reviewers have been addressed and the suggestions have been implemented in our best endeavor. In what follows, the reviewers' comments are reproduced in italics, followed by our responses, which include the associated changes made in the revised manuscript. All reference numbers used below refer to the updated reference list of the revised manuscript unless explicitly stated otherwise.

Response to comments from Reviewer 1:

General comment: The paper is written well and the structure is appropriate.

Answer: We thank the reviewer for the appreciation of the presentation and organization of this paper.

Q1. Composite field S-boxes are widely used and a very important reference is provided below; please explain in the paper the differences between choosing polynomial basis, normal basis, or mixed basis composite fields. Also, elaborations are needed on different subfields and their effects on the complexity of your structures in normal basis and mixed basis. As authors know, subfields in composite fields could be tower field $[GF(((2)^2)^2)^2]$ or semi/half tower field. According to the irreducible polynomials, there are a number of different choices and authors need to either qualitatively explain the effects in replacing their choice with these or do an exhaustive search:

- Mixed Bases for Efficient Inversion in $F((22)2)2$ and Conversion Matrices of SubBytes of AES, Yasuyuki Nogami, Kenta Nekado, Tetsumi Toyota, Naoto Hongo, Yoshitaka Morikawa, CHES 2010.

Answer: We thank the reviewer for the suggestion of a very important reference on mixed bases for SubBytes transformation of AES. This reference has been added as [24] and cited in our revised manuscript. We have carefully read the recommended paper and performed an exhaustive search on different choices of the composite field basis used in SubBytes and InvSubBytes transformations. We acknowledge the validity of the reviewer comment on the differences between polynomial basis, normal basis and mixed basis. Their different multiplicative inversions can impact the complexity of the structures of SubBytes and InvSubBytes blocks of AES if they are implemented by combinational logic circuits. For Block RAM and LUT (ROM) based implementations, the hardware complexity of the SubBytes and InvSubBytes transformations are independent of the way the S-box and inverse S-boxes are computed. As this research addresses in-memory computing architecture, the S-boxes in our

design are realized by LUTs instead of combinational logic. Hence the same proposed in-memory computing architecture can also be applied to S-boxes designed by composite fields represented in different basis. The mappings to the S-box data can be pre-computed and stored in DW-LUT.

The following statements have been added in Page 6, Section IV-B to clarify this concern on the different choices of basis for the composite field S-box.

“The composite fields for S-box and inverse S-box operations can be represented using normal basis, polynomial basis or mixed basis [24]. In the proposed domain-wall nanowire architecture, DW-LUT instead of combinational logic is used to store the transformed data of SubBytes and InvSubBytes, hence the hardware complexity of their implementation is not affected by using different bases for the mapping to composite field S-boxes.”

Q2. Side-channel attacks related to hardware are very relevant when new implementations are presented. Authors have not added explanations on how the new architecture can thwart fault analysis and power analysis attacks. In this regard, the steps include referring to the relevant previous works, for instance, the followings on fault analysis attacks and detail on how the new architecture can counteract as such (references below are some of the relevant previous works):

- R. Karri, K. Wu, P. Mishra, and K. Yongkook, "Fault-based side-channel cryptanalysis tolerant Rijndael symmetric block cipher architecture," in *Proc. DFT*, Oct. 2001, pp. 418-426.
- G. Bertoni, L. Breveglieri, I. Koren, P. Maistri, and V. Piuri, "Error analysis and detection procedures for a hardware implementation of the advanced encryption standard," *IEEE Trans. Computers*, vol. 52, no. 4, pp. 492-505, Apr. 2003.

Answer: We thank the reviewer for this suggestion and the recommendation of two insightful references on fault-based side-channel cryptanalysis on AES. We have carefully studied the existing fault analysis and power analysis based side-channel attacks that have been proven effective on conventional CMOS-AES implementation and came to the following conclusions. To succeed in fault analysis, the attacker must be able to inject an error into a logic operator or net precisely to exploit its altered output state for secret key determination. To succeed in the power analysis, the attacker must be able to measure the power dissipated in different time-slots of encryption. In both attacks, enough information must be acquired to derive a fault or power model of the device in order to correlate a small number of these measurements with each bit of the internal stage of computation during encryption. From these perspectives, both the device (domain wall nano-wire) characteristics and architectural (in-memory computing) properties featured by our proposed DW-AES offer an inherent resistance against these attacks. As the nonlinear byte substitution mappings are replaced by information stored in MTJ and the logic operations are performed within the memory, the attacker cannot inject a fault precisely like a voltage bump in CMOS-AES. To inject a fault, the attacker may apply a strong external magnetic field to influence the magnetic devices. Due to the size of the nanowire, it will be difficult to focus the field into the target nanomagnet to depin the domain wall. The fault injection is imprecise as the inability to localize the external field will easily influence other cells near the target cell. Even though the attacker can inject the same magnetic field for many times,

its influence on the MTJ device is different for each application. The advantages of spin-based instead of charge-based memory and computations are their virtually zero leakage power (normally off) and ultra-low dynamic power consumption, which make it difficult to measure the differential power accurately. Moreover, as we design the logic computation of AES in memory, several ciphers can be running in parallel at the same time. As a result, it is hard to detect the power of a single cipher. Therefore, power analysis based side-channel attack is not feasible.

The following paragraph has been added under a new subsection, Section IV-H “Immunity to Fault and Power Analysis Attacks” in Page 8 of the revised manuscript to elaborate the inherent resistance of the proposed architecture to side-channel attacks. The suggested references have also been added and cited as [28] and [29] in this paragraph.

“Fault analysis and power analysis based side-channel attacks are the major threats in CMOS AES implementations [28], [29]. Our proposed architecture is inherently resilient against these two attacks owing to the use of spintronic devices and in-memory computing architecture. For the fault analysis attack, an external magnetic field is required to inject the fault because the logic and storage are implemented by magnetic materials. Due to the size of the nanomagnets, it is impossible to localize the applied field to influence only a single MTJ or a DW nanowire within a densely integrated magnetic memory array to precisely inject a fault into a target cell. As opposed to charge-based devices, spin-based devices are normally off, they have zero leakage and ultra-low switching power. Moreover, as all the AES logic are implemented inside the memory, it is difficult to correlate the indistinguishable power dissipation of each logic block measured in different time slots of encryption.”

Q3. *Again, other variants of the AES implementations need to be considered. You have to explain how your architecture would affect the AES-GCM architecture, see below for a previous work:*

– A. Satoh, T. Sugawara, and T. Aoki, "High-Performance Hardware Architectures for Galois Counter Mode," *IEEE Trans. Computers*, vol. 58, no. 7, pp. 917-930, July 2009.

Answer: We thank the reviewer for bringing the AES-GCM architecture variants to our attention. We have carefully studied the different AES-GCM architectures described in the recommended paper, (which has been added and cited as [27] in our revised manuscript). According to the block diagram of AES-GCM, a Galois Field (GF) multiplier-adder is the main operation needed to realize GCM on AES. The GCM hash function repeats GF multiplications and additions to produce an authentication tag along with the AES encrypted ciphertext. The 128-bit input to the GCM hash function, which is either authenticated data A or the 128-bit AES ciphertext C , is multiplied with the 128-bit constant hash key H , and the 128-bit products are accumulated in a register to obtain an intermediate hash value. The multiplication over GF can be carried out by a 128×128 AND-array and the summation of partial products can be performed by an XOR-array in a pipelined AES. We have added a new figure (Fig. 10) in the revised manuscript to show that this GCM hash function can be easily realized in our proposed AES architecture by leveraging on the simplicity of ANDing in DW nanowire shifter. As shown in Fig. 10, we can first write the input, which is either A or C , into the DW nanowire, and then use the hash key bits to control the DW shifting. If the hash key bit is 0, there will be no shifting and

the readout of MTJ will be 0. If the data bit is 0, the readout of MTJ will also be 0 irrespective of whether a shifting is required or not. As a result, the circuit in Fig. 10 can perform the DW-AND operation required for the AES-GCM architecture while the summing up of the partial product bits in an XOR-array can be realized by the DW-XOR structure similar to the AddRoundKey stage shown in Fig. 8.

The following paragraph has been added in a new Section IV-G “DW-AND Operation for GCM Architecture Extension” on Page 8 of the revised manuscript to illustrate how the additional required operation of GF hash function for AES-GCM can be realized by incorporating the DW-AND into the domain-wall nanowire shifter shown in the newly added Fig. 10.

“Among different modes of operation for symmetric key cryptographic block ciphers, Galois Counter Mode (GCM) has been widely adopted due to its efficiency and performance. As an efficient authenticated encryption algorithm, AES-GCM [27] can fully exploit the parallel and pipelined processing of the original AES in Figure 1 to provide both data authenticity and confidentiality. Besides all the operations required in Figure 1, a GCM hash function is needed to generate the authentication tag. The Galois Field (GF) multiplication of the GCM hash function can be obtained by ANDing the authenticated data A and the ciphertext data C with a hash key H , where A , C and H are all 128-bit data. This 128×128 AND-array can be implemented using a DW memory based AND operation. Figure 10 shows the detailed implementation of a 1-bit AND operation on the DW nanowire. To perform an AND operation, we need to pre-write a 0 into a domain of the DW nanowire first, and then write a bit of A or C into the next domain. The corresponding bit in H is then applied on the shift transistor. If the hash key bit is 0, the DW nanowire will not shift and the pre-write 0 will be read out. If the data in A or C is 0, the readout data will always be 0 irrespective of whether there is any DW movement in the nanowire. Altogether 128 pairs of 1-bit AND DW nanowire are needed. The GF sum of the partial product bits can be easily realized by the DW-XOR structure as in the AddRoundKey stage shown in Figure 8. Hence the GCM operation can be fully integrated into the DW nanowire based AES.”

Response to comments from Reviewer 2:

General Comments: *The paper presents a block-level in-memory architecture for AES. According to the paper, mass data encryption in a processor creates a bottleneck for low-speed I/O. In that respect the in-memory encryption will facilitate the encrypted data processing. As in this case, the processor will only deal with the data processing rather than encryption. Further, the proposed in-memory encryption if applicable will have huge benefit in terms of area, power, and throughput. The paper has a good structure and satisfactory discussion and results. There are some comments as follows:*

Answer: We thank the reviewer for a concise summary of our contributions and the appreciation of the organization, discussions and results presented in this manuscript.

Q1. *The paper well explained how in-memory AES can be implemented. However, it is not clear how the in-memory AES can be interfaced with the processor to enhance the performance.*

Answer: We thank the reviewer for the meticulous review. Yes, the in-memory AES will still need to have some minimum interaction with the processor. Unlike the conventional architecture, the data for processing is loaded from the storage block to the logic block, both are reside within the memory of

the proposed architecture. Consequently, the processor will only need to issue some commands to the memory for the data transmission. To execute these commands, a control bus is required in the memory for the communication among storage, logic and processor. In addition, some new commands are required because the memory-to-memory data transmission is not required in conventional architecture but necessary in DW-AES. As the interface logic for logic-in-memory architecture is not a new contribution in this work, we refer the readers to our recent published work (reference [19] of the revised manuscript) for the same interface we adopted for the communication between the in-memory AES and the processor.

The following statements have been added in Section II-B on Page 3 of the revised manuscript to clarify the control bus and communication protocol adopted by our proposed in-memory AES.

“Although the data transmission between memory and processor has been substantially reduced by the in-memory architecture, there is still a need to interface with the processor for the transmission and execution of new instructions dedicated to in-memory computing. In the proposed architecture, we have adopted the control bus and communication protocol described in [19] so that very few data transmissions between processor and memory are required for the in-memory AES.”

***Q2.** The comparison in the paper is assumed to be superficial. For in-memory implementation it is no-doubt that a dedicated memory element will be used. It is not clear from the paper how the area overhead is calculated. Section I-C should clearly explain this instead of referring to [4] and [5], as this is one of the major criteria based on which the novelty of the paper is judged.*

Answer: We thank the reviewer for this suggestion. We believe the reviewer refers to Section VI-C instead of Section I-C and references [4] and [6] instead of [4] and [5] in this comment. We would like to clarify that for a fair comparison with our proposed design in 32 nm technology, technology scaling effects have been factored into the excerpted measurement results reported in [4] and [6] for these two existing CMOS-ASIC implementations of AES in Table III. The area estimated for our design is compared against the areas of [4] and [6] mapped to the same technology node in order to estimate the area savings. Since our proposed DW-AES architecture is composed of both CMOS and domain-wall nanowire, the areas consumed by both parts are calculated separately before they are combined to obtain the final area of our design. DW-CACTI, which is developed based on CACTI tool [32], is used to calculate the cell area constituted by the MTJ and domain-wall nanowire devices. In DW-CACTI, the number of CMOS transistors, the size of MTJ and domain-wall nanowire, and the size of the memory array (e.g., the size of DW-LUT) are needed as input for the calculation. To clarify this, an example on how we can calculate the area of $78 \mu\text{m}^2$ of the baseline DW-AES is shown. The SubBytes stage requires a LUT with $2k$ registers and two 4-16 decoders. As mentioned in Section VI-A, the size of each domain is $32 \text{ nm} \times 64 \text{ nm}$. Hence the area of the domain-wall nanowire is $4.2 \mu\text{m}^2$. The area of the two CMOS-based decoders is found to be $0.95 \mu\text{m}^2$. Because the domain-wall nanowire and CMOS are fabricated in different layers, the area of the CMOS are fully subsumed under the domain-

wall nanowire stack. Thus the area occupied by one SubBytes stage is $4.2 \mu\text{m}^2$. All other stages can be calculated in the same manner as SubBytes and their areas are listed as follows:

SubBytes	ShiftRows	MixColumns	AddRoundKey	Area for one cipher	Number of ciphers	Total area
$4.2 \mu\text{m}^2$	$0.04 \mu\text{m}^2$	$3.3 \mu\text{m}^2$	$0.3 \mu\text{m}^2$	$7.8 \mu\text{m}^2$	10	$78 \mu\text{m}^2$

For pipelined DW-AES and multi-issue DW-AES, additional register arrays for the pipeline and hardware resources (e.g., 3 MixColumns in multi-issue DW-AES) are needed. Hence their areas are larger than the baseline DW-AES.

The following statements have been added in Section VI-C-2 on Page 12 to explain the area overheads of DW-AES.

“As mentioned earlier, the areas reported in [4], [6] and [7] are technologically scaled to 32 nm technology by a factor of $1/S^2$. The areas of our proposed architectures are calculated based on the number of CMOS transistors, and the sizes of MTJ, DW nanowire and memory array using the DW-CACTI tool developed from CACTI [32]. As the in-memory architecture is composed of CMOS and DW nanowire circuits, the area consumed in each part is calculated separately. For example, the DW-LUT in Figure 6 requires a 256-byte DW memory and two 4-16 decoders. With each domain occupying an area of $32 \text{ nm} \times 64 \text{ nm}$, the areas of the DW memory and the two decoders are calculated to be $4.2 \mu\text{m}^2$ and $0.95 \mu\text{m}^2$, respectively. Because the domain-wall nanowire and CMOS circuits are implemented in different layers, the much smaller CMOS circuit area is fully subsumed under DW nanowire stack. Hence, the SubBytes stage is dominated by the larger nanowire circuit area of $4.2 \mu\text{m}^2$. The areas of ShiftRows, AddRoundKey and MixColumns are calculated in the same way to be $0.04 \mu\text{m}^2$, $3.3 \mu\text{m}^2$ and $0.3 \mu\text{m}^2$, respectively. As the total area of each encryption cipher is only $7.8 \mu\text{m}^2$, 10 sets of encryption ciphers are implemented in our design. Pipelined DW-AES and multi-issue DW-AES occupy larger areas than the baseline DW-AES due to the additional register arrays for pipeline and hardware resources like 3 MixColumns for multi-issue.”

Q3. *Again, it is not clear how the throughput is enhanced specially when the proposed implementation is of 30MHz speed.*

Answer: We thank the reviewer for this suggestion. After some careful consideration, we found that the term “throughput” can be misleading for the comparison of data rate of different implementations with different number of parallel ciphers and multi-cycle pipelined stages in Table IV. What we intended to compare in Table IV is actually the amount of data that can be encrypted within the same amount of time and area. For a fair comparison, power density is also taken into consideration in view of the number of ciphers that can be executed concurrently within the small footprint for custom hardware implementations. Hence the operating frequency of all hardware implementations under the same 2 mm^2 area budget, including [4], [6] and [7] is set to 30MHz to avoid the over-heating problem in memory. In this context, a more appropriate figure of merit called the data processing rate (DPR) is defined. $\text{DPR} = (\text{number of ciphers} \times \text{clock rate} \times \text{number of bytes per cipher}) / \text{number of cycles in the}$

longest (pipelined) stage of each cipher. It has the unit of Giga Bytes per second. For a 128-bit cipher, the number of bytes per cipher is equal to 16. The number of parallel ciphers that can be implemented within the given area and the number of cycles in the critical stage in each cipher differ from one hardware implementation to another. Therefore, Table IV has been revised to include the number of cycles in the critical stage of each cipher and DPR.

The following statements have been added in Section VI-D on Page 13 to introduce the data processing rate for the comparison of all implementations.

“For more compact implementation such as [6], more AES ciphers can be housed within the 2 mm² area budget.”

“Due to the number of concurrent encryptions within a small footprint, 30MHz is a reasonable clock rate according to Fig. 15(b) for all custom circuit implementations to avoid overheating. Given a uniform operating frequency, the number of cycles for the most time-critical stage is different among all implementations, which is indicated in Table IV for a 128-bit cipher. A figure of merit, called the data processing rate (DPR), is defined to measure the rate of encrypted data within a given area budget for different custom implementations. DPR, expressed in bytes per second, is defined as:

$$DPR = \frac{\#ciphers \times f \times \#bytes \text{ per cipher}}{latency \text{ of critical stage}} \quad (12)$$

where #ciphers is the number of ciphers that can be implemented in a given area, f is the clock frequency in Hz, #bytes per cipher is the number of bytes in each cipher and the latency of critical stage is the number of clock cycles consumed in the longest stage of the AES implementation.”

“Due to the smaller area per cipher of DW-AES, more ciphers can operate in parallel under the same area budget, leading to its higher DPR.”

“This further improvement in DPR is achieved by reducing the latency of the critical stage.”

Q4. *There is a typo at Section IV -B nonlinear transformation=> nonlinear*

Answer: As suggested, the typo in Section IV-B has been corrected.