Energy Efficient In-Memory AES Encryption Based on Nonvolatile Domain-wall Nanowire

Yuhao Wang*, Hao Yu*, Dennis Sylvester†, Pingfan Kong*,
*School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798
†Department of Electrical Engineering and Computer Science, University of Michigan, USA

***Abstract*— The widely applied Advanced Encryption Standard (AES) encryption algorithm is critical in secure big-data storage. Data oriented applications have imposed high throughput and low power, i.e., energy efficiency (J/bit), requirements when applying AES encryption. This paper explores an in-memory AES encryption using the newly introduced domain-wall nanowire. We show that all AES operations can be fully mapped to a logic-in-memory architecture by non-volatile domain-wall nanowire, called DW-AES. The experimental results show that DW-AES can achieve the best energy efficiency of 24 pJ/bit, which is 9X and 6.5X times better than CMOS ASIC and memristive CMOL implementations, respectively. Under the same area budget, the proposed DW-AES exhibits 6.4X higher throughput and 29% power saving compared to a CMOS ASIC implementation; 1.7X higher throughput and 74% power reduction compared to a memristive CMOL implementation.**

I. Introduction

Due to instant-on power-up and ultra-low leakage power, the newly introduced nano-scale non-volatile memory (NVM) such as ReRAM [1] and STT-RAM [2] has shown great potential for future big-data storage. However, the sensitive data will not be lost during reboot or suspension and hence is susceptible to attack. Further, large volume of data must be encrypted with high throughput and low power. Traditional memory-logic integration based design incurs large overhead when performing encryption by logic through I/Os. Therefore, in-memory encryption would be preferred to achieve high energy efficiency during data protection.

Various CMOS-based hardware implementations for AES have been presented [3]. In scenarios where energy efficiency is critical, CMOS-based ASIC implementations tend to incur significant leakage power in current deep sub-micron regime with limited throughput. In [4], a memristive CMOL implementation by hybrid CMOS and ReRAM design is introduced to facilitate AES application. However, while the ReRAM serves as reconfigurable interconnection, it is not used for in-memory computation based encryption.

As spintronic devices have shown great scalability, it is promising to build big-data storage with in-memory logic based computing such as encryption. Domain-wall nanowire [5], [6], or racetrack memory, is a newly introduced spintronic NVM device. It has not only potential for high density and high performance memory design, but also interesting in-memory computing capability [7], [8] compared to other emerging NVM technologies. In this work, we propose a full domain-wall nanowire device based in-memory AES computing, called DW-AES. The non-volatile domain-wall nanowire devices are both used as storage element and deployed for logic computing in AES encryption. Experimental results show that the proposed DW-AES outperforms both CMOS-based ASIC and the hybrid CMOS/ReRAM based AES computing for data storage. Respectively, energy efficiency is improved by 9X and 6.5X, and throughput improves by 6.4X and 1.7X.

The rest of the paper is organized as follows. Section II details how all AES transformations can be implemented by the domain-wall nanowire. Experiment results are presented in Section III with conclusion in Section IV.

978-3-9815370-2-4/DATE14/©2014 EDAA

Please address comments to haoyu@ntu.edu.sg. This work is sponsored by Singapore NRF-CRP(NRF-CRP9-2011-01) and MOE Tier-2 (MOE2010-T2-2-037 (ARC 5/11)). The authors would also like to express their sincere appreciation to Chuliang Weng, Junfeng Zhao, and Zhulin Wei from Shannon Laboratory, Huawei Technologies Co., Ltd, China for useful discussions.

II. Domain-wall Nanowire Based AES Computing

In-memory encryption offers two major advantages over existing approaches. Firstly, all domain-wall based AES ciphers (DW-AES) can be integrated inside the memory, and AES encryption is performed directly on target data stored in non-volatile domain-wall memory. This is significantly different from the conventional memory-logic architecture in which the non-volatile storage data to process must be loaded into volatile main memory, processed by logic, and written back afterwards. Secondly, the DW-AES cipher is implemented purely by domain-wall nanowire devices, which are identical to the storage elements. This provides good integration compatibility between DW-AES ciphers and the memory elements, as well as the ability to reuse peripheral circuits like decoders and sense amplifiers. In this section, the detailed domain-wall nanowire based in-memory encryption will be discussed.

A. Data Organization of State Matrix

In AES, the standard input length is 16 bytes (128 bits), which are internally organized as a two-dimensional four rows

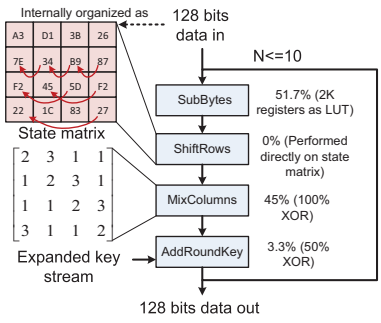

Fig. 1: The flow chart of AES algorithm with gate utilization analysis

by four columns array, called *state matrix*. During the AES algorithm, a sequence of transformations are applied to the state matrix, after which the input block is considered encrypted and then output. The flow chart of the AES algorithm is shown in Figure 1. Because in-memory encryption is performed directly on data cells, the data needs to be organized in certain fashion to facilitate the AES algorithm.

As domain-wall nanowires only support serial access, the data needs to be distributed into separate nanowires so that multiple bits can be operated concurrently within one cycle. In AES algorithm, the basic processing unit is each byte in the state matrix. Therefore, the state matrix is split into eight 4×4 arrays, as illustrated in Figure 2, where each entry of each array becomes one bit instead of one byte. By distributing the bytes and operating eight arrays together, the byte access requirement in AES algorithm is satisfied. Moreover, to facilitate the ShiftRows transformation by exploiting the shift property of domain-wall nanowire, each row of an array needs to be stored within one domain-wall nanowire. In this case, each array is composed of four nanowires, and within each nanowire, the four bits data are kept along with some redundant bits used for efficient circular shift. Details regarding redundant bits will be discussed later in ShiftRows transformation. By organizing each 16 bytes of data in the above manner, the AES algorithm can be applied efficiently.

B. AddRoundKey

In the AddRoundKey step, each byte in the state array will be updated by bit-wise XOR with corresponding key byte. As the dominant operation in this step is XOR, shown in Figure 1, we propose a nanowire based XOR logic (DW-XOR) for leakage free computing. As the GMR-effect in the two magnetic layers structure can be interpreted as the bitwise-XOR operation of the magnetization directions of two thin magnetic layers, where the output is denoted by high or low resistance. In a GMR-based MTJ structure, however, the XOR-logic will fail as there is only one operand as variable since magnetization in the fixed layer is constant. This problem is overcome by the unique domain-wall shift-operation in the domain-wall nanowire device, which enables DW-XOR for computing.

The AddRoundKey with bitwise-XOR logic implemented by two domain-wall nanowires is shown in Figure 3. The

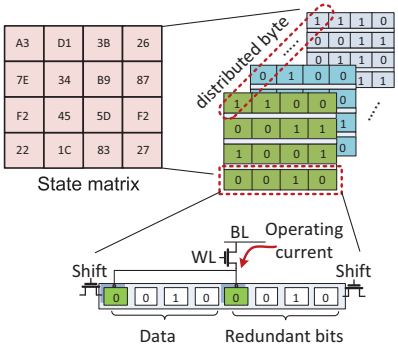

Fig. 2: Data organization of state matrix by domain-wall nanowire devices in distributed manner

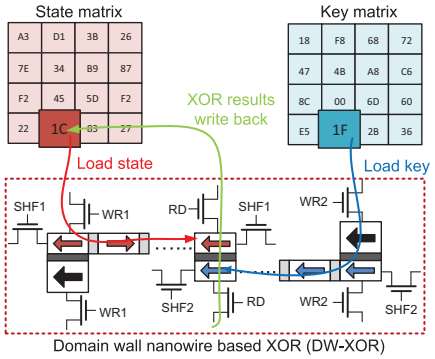

Fig. 3: AddRoundKey step with XOR logic achieved by domain-wall nanowire

proposed bitwise DW-XOR logic is performed by constructing a new read-only-port, where two free layers and one insulator layer are stacked. The two free layers each have the size of one magnetization domain and are from two respective nanowires. Thus, the two operands, representing the magnetization direction in each free layer, can both be variables with values assigned through the MTJs of their own nanowire. These assigned values are then shifted to the operating port such that the XOR can be performed by detecting the resistance.

C. SubBytes

In this step, each byte in the state matrix will undergo an invertible non-linear transformation. This transformation is commonly implemented as a look-up table (LUT), called substitution box (S-box). S-box LUT, essentially a pre-configured memory array, takes 8 bit input as a binary address, finds target cells that contain 8 bit result through decoders, and finally outputs correspondingly by sense amplifiers. With 2^8 possible input scenarios, and each scenario having 8 bit result, the LUT size can be determined as $2^8 \cdot 8 = 2048$ bits. The LUT is conventionally implemented by SRAM cells, which in this size will incur significant leakage power. Readily implemented by domain wall nanowire device, the DW-LUT will enable significant leakage reduction, In addition, the memory and DW-LUT can share decoders and sense amplifiers, which leads to further power and area savings.

D. ShiftRows

The ShiftRows transformation can be efficiently achieved by exploiting the unique shift property of domain-wall nanowire. Due to the distributed data organization, in the ShiftRows transformation, the second row needs to be left shifted cyclically by one bit, the third by two bits, and the fourth row by three bits, while the top row remains unshifted. In order to accomplish the circular shift in an elegant manner, i.e. without writing back the most significant bits to the least significant bits, redundant bits are used to form a virtual circle on the nanowire, as illustrated in Figure 4.

Since each row has predetermined shift operation, the number of redundant bits of each row can be readily determined: one redundant bit is required for second row, two bits for third row, and three bits for last row, all attached to the least significant bit from right side In order to achieve all shifts in one cycle, shift currents of different amplitude are applied to

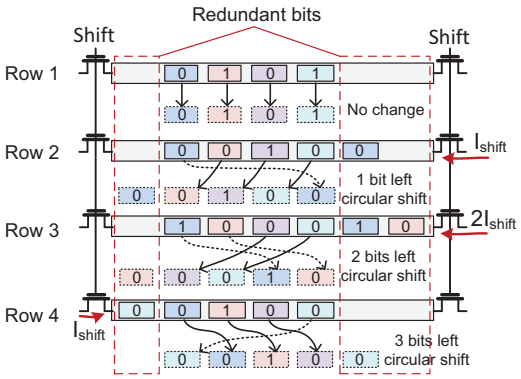

Fig. 4: ShiftRows transformation by domain-wall nanowire shift operations

each row according to the linear current-velocity relationship of shift operation [9]. In other words, the third row and fourth row are applied shift current that is twice and three times the amplitude applied to the second row. Consider the equivalent operations in circular shift with four bits: *LS1* $\stackrel{\text{def}}{=\!=}$ *RS3*, *LS2* $\stackrel{\text{def}}{=\!=}$ *RS2*, *LS3* $\stackrel{\text{def}}{=\!=}$ *RS1*, where *LS* and *RS* indicate left and right shift, the number denotes the length to shift. This means in last row, instead of shifting 3 bits leftward, only right shift 1 bit needs to be performed. This helps to reduce the redundant data from 3 bits to 1 bit, as well as reduce the applied shift current to one third of previously required amplitude. The bits in same color indicate that they are synchronized bits. To ensure correct circular shift, the redundant bits need to be synchronized with their counterparts. As a result, during changes in the matrix state the redundant bits must also be updated. In contrast with conventional computing flow, in which data needs to be moved to computing units for execution and written back to memory afterwards, the ShiftRows transformation is done directly on the stored data by in-memory computing fashion.

E. MixColumns

The MixColumns transformation can be expressed as the state matrix multiplied by the known matrix shown in Figure 1. The operations needed are multiplication by two (*xtime-2*), multiplication by three (*xtime-3*), and addition defined as bit-wise XOR. The *xtime-2* is defined by left shift by 1 bit, and bit-wise XOR with *0x1B* if the most significant bit is 1; The *xtime-3* is defined as *xtime-2* result XOR with its original value. Therefore, there are only two de-facto atomic operations: 1) bit-wise XOR, executed by proposed DW-XOR, and 2) *xtime-2*. Although *xtime-2* can be implemented by in-memory shift together with additional DW-XOR, it is more efficient to use 8-bit input 8-bit output DW-LUT due to its branch operations depending on its most significant bit. As such, the MixColumns transformation can be purely performed by DW-LUT and DW-XOR.

III. Experimental Results

A. Experiment Setup

To evaluate DW-AES cipher, the following design platform has been set up. Firstly at device level, the transient simulation of MTJ read and write operations are performed within NVM-SPICE [10], [11], [12], [13], [14] to obtain accurate operation energy and timing for domain-wall nanowire. The shift-operation energy is modeled as the Joule heat dissipated on the nanowire when shift-current is applied. The shift-current density and shift-velocity relationship are based on [9]. The area of one domain-wall nanowire is calculated by its dimension parameters. Specifically, the technology node of 32nm is assumed with width of 32nm, length of 64nm per domain, and thickness of 2.2nm for one domain-wall nanowire; the R_{off} is set at 2600Ω, the R_{on} at 1000 Ω, the writing current at $100\mu A$, and the current density at $6 \times 10^8 A/cm^2$ for shift-operation. Secondly at circuit level, the memory modeling tool CACTI [15] is modified with name as DW-CACTI. It can provide accurate power and area information for domain-wall nanowire memory peripheral circuits such as decoders and sense amplifiers (SAs). Together with the device level performance data, the DW-XOR as well as the DW-LUT can be evaluated at circuit level. The additional sequential controller of DW-AES is described by Verilog HDL, which is synthesized with area and power profiles. Finally at system level, an AES behavioral simulator is developed to emulate the AES cipher, as well as to explore the trade-offs among power, area, and speed.

B. AES Performance Comparison

The proposed DW-AES cipher is compared with both CMOS-based ASIC design [3] and hybrid CMOS/ReRAM (CMOL) design [4]. For these implementations, performance data is extracted from the reported results in [3], [4] with necessary technology scaling. C-code based software implementation that runs on a general purpose processor (GPP) is also compared. Evaluation of the AES software implementation is done in two steps. Firstly, gem5 [16] simulator is employed to take AES binary, compiled from C-code obtained from [17], which generates the runtime utilization rate of core components. Next, the generated statistics are taken by McPAT [18], which provides core power and area model. All hardware implementations run at the clock-rate of 3MHz, while the processor is operated at 2GHz for the software implementation. Table I compares the different implementations of AES cipher, and the results are discussed as follows.

TABLE I: AES for 128 bits encryption performance comparisons

Implementation	leakage (μW)	total power power(μW)	area (μm^2)	cycles
C code [17] on GPP	1.3e+6	4e+5	2.5e+6	2309
CMOS ASIC [3]	120.54	154.74	953.05	534
memristive CMOL [4]	102.35	119.04	251.5	534
DW-AES	14.602	21.568	78.121	1022

As expected, the DW-AES cipher has the smallest leakage power due to the use of non-volatile domain-wall nanowire devices. The remaining small leakage power is introduced by its CMOS peripheral circuits, i.e. decoders, sense amplifiers, as well as simple sequential controllers. Specifically, DW-AES cipher achieves a leakage power reduction of 88% and 86% compared to the CMOS ASIC and memristive CMOL designs, respectively. The leakage power can be further reduced if the

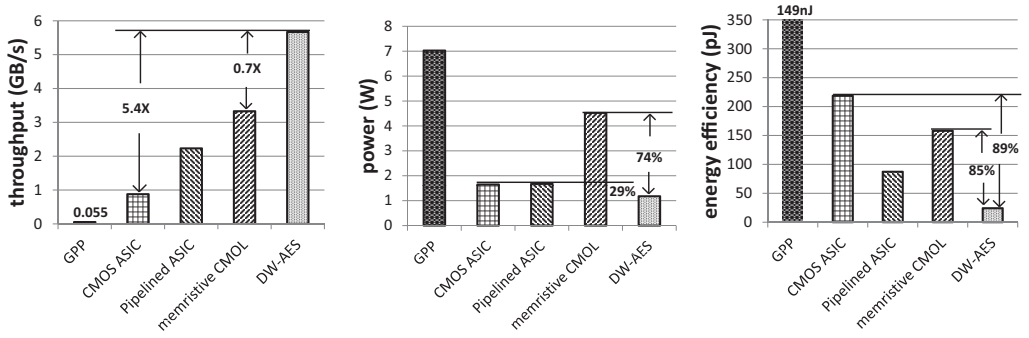

Fig. 5: In-memory encryption throughput, power and energy efficiency comparisons between different AES platforms

decoders and SAs of the memory can be reused by the DW-AES ciphers.

The tradeoff made in the DW-AES cipher is a larger number of cycles required compared to other hardware implementations. This is caused by the multiple-cycle operations of DW-XOR and its DW-LUT, where the shift-operation needs to be performed first in order to align the target cell with MTJ to operate. Note that while small latency between the raw data in and the encrypted data out is critical in real-time systems, in big-data applications the most significant figures of merit are throughput and energy efficiency.

C. Throughput and Energy Efficiency Comparison

In the following, 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 subject to a given $10mm^2$ area constraint. All AES units are encrypting input data stream concurrently due to the high data parallelism. With the exception of the proposed in-memory DW-AES, all platforms will incur I/O energy overhead of 3.7nJ per memory access. For memory, a capacity of 1GB and buswidth of 128 bits are assumed.

Figure 5 compares throughput, power, and energy efficiency of different AES computing platforms. All AES hardware implementations have several orders of magnitude throughput and energy efficiency improvement compared to the software implementation on general purpose processor, as expected. Among all the hardware implementations, the proposed DW-AES computing platform provides the highest throughput of 5.6 GB/s. This throughput is 6.4X higher than that of the CMOS ASIC based platform with a power saving of 29%; 2.5X higher than that of the pipelined CMOS ASIC platform with 30% power reduction; and 1.7X times higher than that of memristive CMOL based platform with 74% power saving. Due to the in-memory encryption computing and non-volatility, the proposed DW-AES computing platform can achieve the best energy efficiency of 24pJ/bit, which is 9X, 3.6X, 6.5X times higher than its counterpart: the CMOS ASIC, pipelined CMOS ASIC, and memristive CMOL based platforms, respectively.

IV. Conclusion

The domain-wall nanowire based AES (DW-AES) is introduced in this paper. All AES operations can be fully mapped to exploit the unique properties of this emerging technology. For example, the DW-XOR logic is proposed for the dominant XOR operation; the domain-wall shift is exploited for the row-shift operation; and the DW-LUT is utilized for the S-box operation. The experiment results show that, the proposed DW-AES exhibits the best energy efficiency (24 pJ/bit). Respectively, it is 9X and 6.5X better than CMOS ASIC and memristive CMOL based platforms.

References

[1] Y. Wang and et al., "Design of low power 3d hybrid memory by non-volatile cbram-crossbar with block-level data-retention," in *Proc. ISLPED*. ACM, 2012, pp. 197–202.

[2] T. Kawahara and et al., "2mb spin-transfer torque ram (spram) with bit-by-bit bidirectional current write and parallelizing-direction current read," in *ISSCC*. IEEE, 2007, pp. 480–617.

[3] J.-P. Kaps and B. Sunar, "Energy comparison of aes and sha-1 for ubiquitous computing," in *Emerging Directions in Embedded and Ubiquitous Computing*. Springer, 2006, pp. 372–381.

[4] Z. Abid and et al., "Efficient cmol gate designs for cryptography applications," *IEEE Trans. on Nanotechnology*, vol. 8, no. 3, pp. 315–321, 2009.

[5] S. S. Parkin, M. Hayashi, and L. Thomas, "Magnetic domain-wall racetrack memory," *Science*, vol. 320, no. 5873, pp. 190–194, 2008.

[6] L. Thomas and et al., "Racetrack memory: a high-performance, low-cost, non-volatile memory based on magnetic domain walls," in *Proc. IEDM*. IEEE, 2011, pp. 24–2.

[7] Y. Wang and H. Yu, "An ultralow-power memory-based big-data computing platform by nonvolatile domain-wall nanowire devices," in *Proc. ISLPED*. IEEE, 2013, pp. 329–334.

[8] Y. Wang and et al., "Energy efficient in-memory machine learning for data intensive image-processing by non-volatile domain-wall memory," in *Proc. ASPDAC*. IEEE, 2014.

[9] C. Augustine and et al., "Numerical analysis of domain wall propagation for dense memory arrays," in *Proc. IEDM*. IEEE, 2011, pp. 17–6.

[10] Nvmspice - spice for non-volatile memory technologies. [Online]. Available: http://www.nvmspice.org/

[11] Y. Shang and et al., "Analysis and modeling of internal state variables for dynamic effects of nonvolatile memory devices," *IEEE Trans. on TCAS-I*, vol. 59, no. 9, pp. 1906–1918, 2012.

[12] Y. Wang and et al., "Spice simulator for hybrid cmos memristor circuit and system," in *Proc. CNNA*. IEEE, 2012, pp. 1–6.

[13] W. Fei and et.al., "Design exploration of hybrid cmos and memristor circuit by new modified nodal analysis," *IEEE TVLSI*, 2011.

[14] Y. Wang and et al., "Design of non-destructive single-sawtooth pulse based readout for stt-ram by nvm-spice," in *Proc. NVMTS*. IEEE, 2012, pp. 68–72.

[15] S. J. Wilton and N. P. Jouppi, "Cacti: An enhanced cache access and cycle time model," *IEEE JSSC*, vol. 31, no. 5, pp. 677–688, 1996.

[16] N. Binkert and et al., "The gem5 simulator," *ACM SIGARCH Computer Architecture News*, vol. 39, no. 2, pp. 1–7, 2011.

[17] Byte-oriented-aes: A public domain byte-oriented implementation of aes in c. [Online]. Available: https://code.google.com/p/byte-oriented-aes/

[18] S. Li and et al., "Mcpat: an integrated power, area, and timing modeling framework for multicore and manycore architectures," in *Proc. MICRO*. IEEE, 2009, pp. 469–480.