# Crypto-DSEDA: A Domain-Specific EDA Flow for CiM-Based Cryptographic Accelerators

**Rui Liu**
School of Materials Science and Engineering
Xiangtan University
Xiangtan 411105, China

Institute of Computing Technology
Chinese Academy of Sciences
Beijing 100083, China

**Zerun Li, Xiaoyu Zhang, Wanqian Li, and Libo Shen**
Institute of Computing Technology
Chinese Academy of Sciences
Beijing 100083, China

University of Chinese Academy of Sciences
Beijing 101408, China

**Rui Tang**
Institute of Computing Technology
Chinese Academy of Sciences
Beijing 100083, China

School of Microelectronics
University of Science and Technology of China
Hefei 230052, China

**Zhejian Luo**
School of Materials Science and Engineering
Xiangtan University
Xiangtan 411105, China

Institute of Computing Technology
Chinese Academy of Sciences
Beijing 100083, China

**Xiaoming Chen and Yinhe Han**
Institute of Computing Technology
Chinese Academy of Sciences
Beijing 100083, China

University of Chinese Academy of Sciences
Beijing 101408, China

**Minghua Tang**
School of Materials Science and Engineering
Xiangtan University
Xiangtan 411105, China

*Digital Object Identifier 10.1109/MDAT.2024.3395987*
*Date of publication: 1 May 2024; date of current version: 29 August 2024.*



Copublished by the IEEE CEDA, IEEE CASS, IEEE SSCS, and TTTC

*Editor's notes:*
The authors propose Crypto-DSEDA, an auto-generation framework for exploring Computing-in-Memory (CiM)-based cryptographic accelerators. Crypto-DSEDA includes an automatic architecture generation pipeline that takes behavioral-level descriptions as input, optimizes dataflow and resource allocation through design space exploration, and finally generates the accelerator architecture and dataflow of the target algorithm.
*—Yiran Chen, Duke University, USA*

**With the development** of the information and digital society, information security and privacy protection have become particularly important, which has also led to extensive research and applications of cryptography. However, traditional encryption methods rely on the CPU-side encryption engine, with data being transmitted between the memory and CPU, resulting in high latency and energy costs [1], [2].

Computing-in-memory (CiM) is a promising technology that can solve the "memory wall" problem. One type of CiM is the in-memory logic performed within memory arrays. By activating two memory rows simultaneously and modifying the sense amplifiers (SAs), in-memory bitwise Boolean logic operations can be realized [3], [4]. Cryptographic algorithms involve a large number of Boolean logic operations, which can be efficiently accelerated by CiM. Some cryptographic accelerators based on CiM architectures have been proposed (e.g., [1] and [2]).

Existing CiM-based cryptographic accelerators are highly customized for the target algorithms, which belong to domain-specific chips. Designing them manually requires experienced developers and a lot of human effort and is nearly impossible to exhaustively explore the design space to comprehensively optimize power, performance, and area (PPA). Therefore, EDA tools are needed to design and optimize the architectures. However, traditional EDA tools are aimed at CMOS-based Boolean logic that can only implement designs below the gate level and lacks application-level information so that they cannot model and optimize domain-specific problems, such as data flow optimization. Furthermore, there are obvious differences between the logic implementations of CiM architectures and traditional CMOS architectures, such as the basic computation implementation, circuit connection relationship, and optimization objectives. New synthesis methods need to be explored to realize the conversion from the upper-level description to logic mapping. In summary, it is meaningful to develop EDA tools for domain-specific CiM chips.

In this work, we propose a domain-specific EDA flow, Crypto-DSEDA, for autogenerating CiM architectures for cryptographic algorithms. According to the algorithm description and design constraints, Crypto-DSEDA generates a CiM-based cryptographic accelerator with optimized PPA. Our contributions to this article are given as follows.

- We define a flexible input method, whereby any formulaic cryptographic algorithm can be explored through the Crypto-DSEDA framework.
- We design a compilation framework for cryptographic algorithms to optimize the data flow of the CiM operations in the iterative process.
- We explore the constructed design space to automatically tailor a CIM-based accelerator for a target algorithm, subject to certain design constraints. The evaluation results show the effectiveness of data flow graph (DFG) optimization, and the Crypto-DSEDA-based accelerator also exhibits considerable performance and energy advantages compared with existing CiM-based implementations.

## Related work

Benefiting from the excellent performance and energy efficiency of CiM architectures, CiM-based cryptographic accelerators have been studied. I-NVMM [5] accelerates the AES algorithm by incrementally encrypting the main memory. IMCSD [6] and AIM [1] further reduce latency and energy by using custom SA circuits. On the basis of AIM, EIM [2] designs new SA circuits and adds the latch operation mechanism, which effectively reduces the write-back operations of intermediate results. FeCrypto [7] proposes an instruction set architecture for cryptographic algorithms based on CiM, which supports multiple cryptographic algorithms.

However, these existing CiM-based accelerators all rely on manual design, and each algorithm requires a customized computing process and architecture. This also means that developing a new accelerator requires a lot of time and labor costs, and even experienced developers cannot fully optimize the PPA of

the architecture. Traditional EDA tools are not fully capable of domain-specific chip design due to the inability to perform application-level optimization, and it is not suitable for synthesizing CiM architectures due to different logic implementations. Therefore, there is an urgent requirement to develop new EDA tools to automatically generate objective-optimal CiM-based cryptographic accelerators.

## Crypto-DSEDA framework

Crypto-DSEDA is an automatic framework for generating CiM-based cryptographic accelerators, which relies on pre-designed basic circuit modules. It works for various CiM architectures that support basic Boolean logic operations. As shown in Figure 1a, the generation of the architecture is mainly divided into four stages: compilation, scheduling, resource mapping, and design space exploration.

Crypto-DSEDA uses an architecture template for architecture generation, as shown in Figure 1b. An accelerator consists of 16 banks, where each bank has eight computing units (CUs). Each CU contains a configurable number of memory arrays, shifters, and lookup tables (LUTs). As a parameterized architecture template, the memory array mainly includes a word-line driver, a sense line driver, an address decoder, SAs and CiM logics, 256×256 memory cells, and a forwarding row (FwRow) [7], where the SAs and CiM logics, the cell structures, and the peripheral module reuse scheme can be customized individually.

### Input structure

The inputs to the framework include a cryptographic algorithm description, design objectives, and constraints. Cryptographic algorithms can be completely described by formulas. Cryptographic algorithms are typically composed of a number of Boolean logic operations, as shown in Table 1. AND, OR, NOT, XOR, IMP, and ADD operations rely on the memory arrays to perform CiM operations. Four types of shift operations are needed: ROL, ROR, SHL, and SHR. The implementation of the MUL operation on the finite field of GF($2^8$) and the nonlinear SubByte operation are implemented by LUTs. Since most hash algorithms mainly modify the compression functions during iterations, to simplify the algorithm description, the structure of the formulas is divided into two parts: 1) the main function and 2) compression

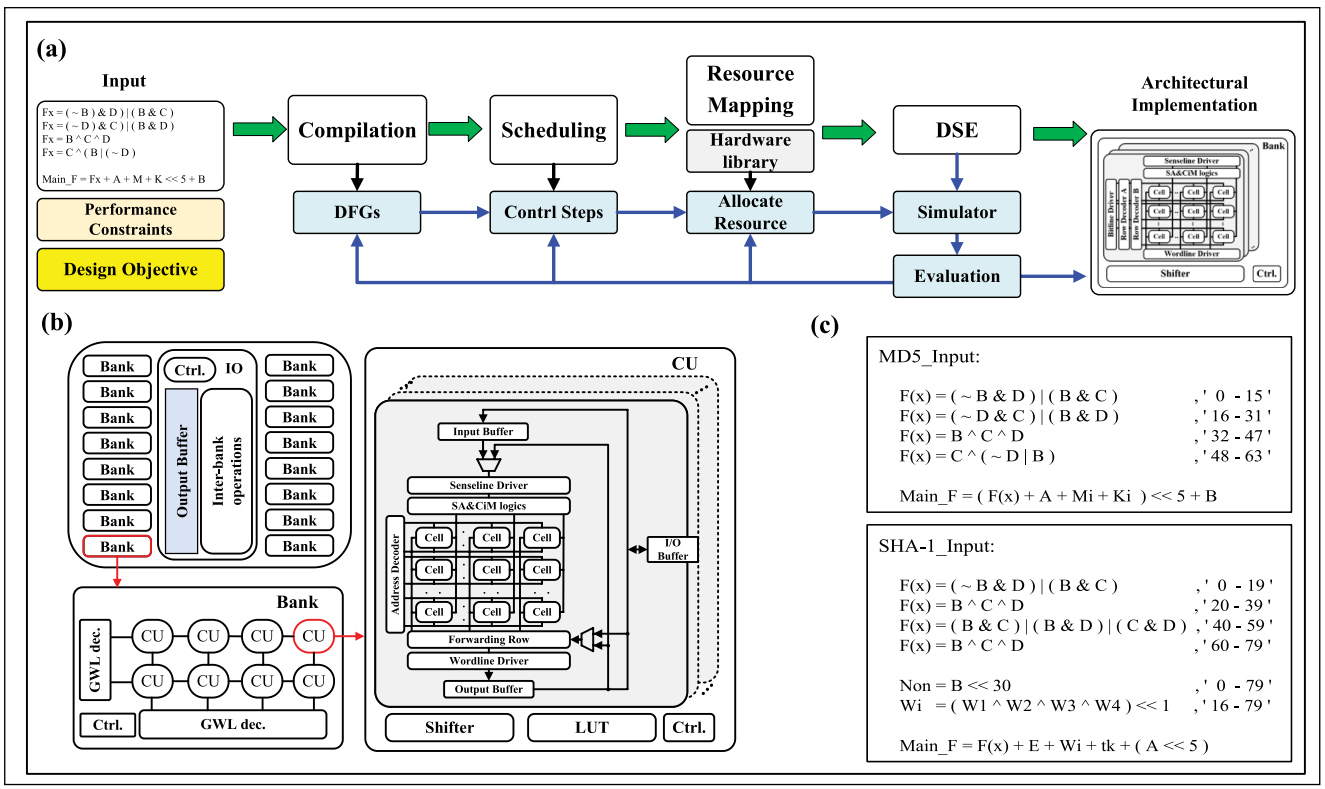


**Figure 1. (a) Overview of Crypto-DSEDA framework. (b) Architecture template. (c) MD5 algorithm and SHA-1 algorithm descriptions in Crypto-DSEDA.**

**Table 1. Operations of cryptographic algorithms.**

| Operation | Symbol | Execution Module |
|---|---|---|
| AND | & | Memory |
| OR | \| | Memory |
| NOT | ∼ | Memory |
| XOR | ∧ | Memory |
| IMP | ∼& | Memory |
| ADD | + | Memory |
| ROL / ROR | ≪ / ≫ | Shifter |
| SHL / SHR | <- / -> | Shifter |
| MUL | ∗ | LUT / Memory |
| SubByte | −− | LUT |

functions and their execution rounds. As two examples, the algorithm descriptions of MD5 and SHA-1 are shown in Figure 1c. Crypto-DSEDA receives these algorithmic formulas and sends them to the compiler to generate an initial DFG.

## Compiler

The compiler receives the cryptographic algorithm description and performs four main tasks: 1) iterative optimization; 2) DFG construction; 3) DFG optimization; and 4) write-back operation insertion.

### Iterative optimization

Cryptographic algorithms have a number of iterations. Since each iteration may only change part of the intermediate variables, read and write operations can be effectively reduced through address remapping, instead of moving unchanged data. As an example, the iterative implementation of SHA-1 is shown in Figure 2a. We only use the operation results of $E$ and $B$ to overwrite the corresponding memory rows and retain other data. After that, the original addresses of $A$, $B^*$, $C$, $D$, and $E^*$ are reassigned to $B'$, $C'$, $D'$, $E'$, and $A'$, respectively, through the compiler and used as the input for the next iteration, achieving an equivalent effect to moving all data.

### DFG construction

The compiler first splits the main function and compression functions into operators and operands. Subsequently, the bonding points between the main function and compression functions are determined. Finally, the compiler will generate the initial DFG based on the bonding points and execution rounds of the compression functions. Figure 2b shows the generation process of SHA-1.

### DFG optimization

DFG optimization consists of two steps: operator overhead detection and redundant operation detection. In operator overhead detection, operators with high hardware overhead or computational overhead will be replaced. The MUL operation on the finite field of GF($2^8$) can obtain results directly through LUTs, but it incurs a high area overhead. In addition, the MUL operation can also be implemented using XOR operations and shift operations to reduce area overhead, but it results in a longer execution time. Furthermore, consecutive NOT and AND operations can be converted to an IMP operation to obtain lower latency and energy, such as "$(\sim B)\&D$." Since most existing CiM architectures do not natively support IMP operations, we need to place corresponding SA modules to implement in-memory IMP operations. Conversely, a smaller memory array area can be obtained by splitting the IMP operation into NOT and AND operations. Redundant operation detection mainly includes two parts. The first part is to eliminate redundant memory access operations. As shown in Figure 2c, operand $B$ needs to perform a shift operation and a NOT operation. Since the operands in the CiM architecture are stored in memory, operand $B$ needs to be read before the shift operation is performed. The read operation is a suboperation of the NOT operation, which also means that operand $B$ can be read out when performing the NOT operation. The second part is to eliminate redundant computing operations. In hash algorithms, the results of each iteration are cyclically shifted and will be used in the next round. As shown in Figure 2a and c, the "$B \wedge C \wedge D$" operation in the round 2 is equivalent to "$A' \wedge B \wedge C$" operation in the round 1. Therefore, we can use the registers to temporarily store the result of "$B \wedge C$" in round 1 to reduce the repeated operations in round 2.

### Write-back operation insertion

Conventional CiM architectures need to write intermediate results back to the original memories when performing continuous operations. To avoid the high latency and energy overhead of writing to nonvolatile memories, we use an FwRow mechanism [7]. Therefore, we only need to perform write-back operations when necessary. Only when both data to be computed are stored in registers, one of the data needs to be written back to memory. In addition, the

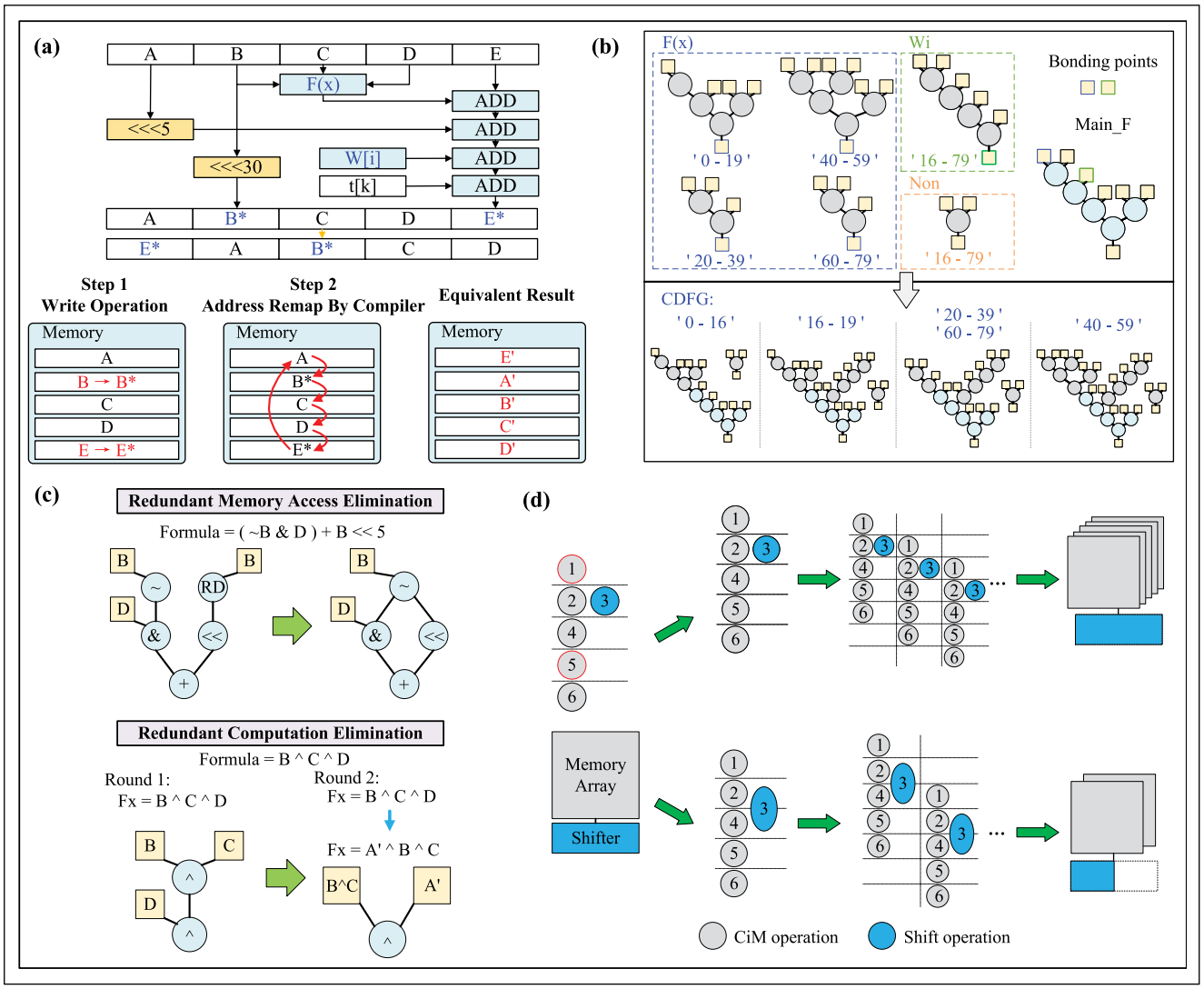


**Figure 2. (a) Iterative implementation of SHA-1. (b) DFG construction of SHA-1. (c) Redundant operation detection. (d) Example for resource allocation.**

results of each iteration will also be written back to memory for the next round of processing.

The DFGs obtained after compilation will be used as input for the scheduling phase.

## Schedule and resource allocation

Operator scheduling in CiM architectures is different from that in traditional CMOS designs, as a memory array can only perform one logic operation at a time. It also means that only one CiM operation can be placed in each control step. Therefore, memories, shifters, and LUTs should be called at the same time as possible to reduce the number of control steps when performing operator scheduling. The minimum number of control steps will be recorded and delete bad results during scheduling.

The recorded schedule results will be used to determine feasible CU configurations. For a scheduled DFG, its resource allocation involves the following two steps. The first step is to calculate the lifetime of shifters and LUTs to determine the maximum reuse factor of peripheral modules, which will be used to confirm the number of peripheral modules. The second step is to count the number of memory arrays using the same set of peripheral modules and determine the offset of their control steps under different reuse factors. Figure 2d depicts an example of hardware resource allocation. Node 3 represents the shift operation, and its parent and child nodes

are Node 1 and Node 5, respectively. Therefore, the lifetime of node 3 is two control steps, and the optional reuse factor is [1, 2]. When the reuse factor is 1, the same set of shifters can be multiplexed up to five memory arrays, and their offsets are [0, 1, 2, 3, 4]. When the reuse factor is 2, node 3 occupies two control steps, CS2 and CS3, and only half a set of shifters need to be placed. Similar to the previous case, by delaying the control step, a single set of shifters can be reused by up to three memory arrays, and their offsets are [0, 2, 4]. It is worth noting that when both shifters and LUTs are placed in the CU, the actual offset of the memory is the intersection of the offsets of shifters and LUTs.

### Design space exploration

Crypto-DSEDA explores different architectural implementations by employing genetic algorithms, where each implementation utilizes a different amount of resources and, therefore, has different latency, energy, and area. The design space is summarized in Table 2. The chromosome of the genetic algorithm is divided into three parts, corresponding to the scheduled DFGs, CU configurations, and hardware parameters, and the encode of each gene is an integer. For example, A chromosome is encoded as [1, 0, 1, 5, 0, 4, 2, 0, 2, 3]. The first four genes are the scheduled data flow part, which specifies the DFGs of each algorithm formula. The fifth and sixth genes are the CU configuration part. The 5th gene specifies the reuse factor combination of shifter and LUT, and the sixth gene specifies the number of memory arrays in each CU. The last four genes determine the hardware parameters of the accelerator, including SAs, memory cells, shifters, and LUTs. In the initialization phase, we randomly generate a population of 300 individuals and calculate their fitness. Subsequently, the next generation population is obtained with the roulette wheel selection. In addition, crossover and mutation are important means to obtain the optimal solution for architecture design. In the crossover phase, two crossover points are randomly marked on two adjacent chromosomes, and then, part of their genes are exchanged. During the mutation phase, the algorithm will randomly mark a gene in the chromosome and randomly pick a new value within its legal range to replace the gene. Finally, extract the optimal chromosome after 1,000 iterations.

**Table 2. Design space of CiM-based cryptographic algorithm accelerators.**

| Design variable | Definition |
|---|---|
| $DFType$ | Combination of scheduled dataflow results |
| $SAType$ | Different configurations of SA and CiM logic circuits |
| $CellStru$ | The memory cell structure, like 1T, 2T, 3T |
| $ShiftType$ | Types of shifters with different parameters |
| $LUTType$ | Types of LUTs with different parameters |
| $ReModul$ | Reusability of shifters and LUTs |
| $MemAlloc$ | Number of memory arrays in each CU |

## Evaluation

We use the ferroelectric field-effect transistor (FeFET) as an example to evaluate Crypto-DSEDA. Before performing DSE, we build a hardware parameter library and presimulate six CiM operations. We construct memory cells and SAs [2], [8], [10] with the 45-nm predictive technology model (PTM) [11] and the FeFET model [12]. Synopsys design compiler is used to synthesize digital logic circuits to obtain their latency, power, and area overhead. We use Crypto-DSEDA to explore several common hash algorithms, namely, MD5, RIPEMD-160, and SHA-1, to evaluate the performance of generated CiM architectures. Finally, we explore the AES algorithm and compare it with a recent CiM-based dedicated AES accelerator, EIM [2], to demonstrate the advantages of Crypto-DSEDA.

Running time: Crypto-DSEDA is implemented by Python. According to the complexity of the algorithm, the exploration time ranges from 10 to 600 s.

### Compiler optimization evaluation

Figure 3a illustrates the results of Crypto-DSEDA exploring each hash algorithm and also shows the exploration results without DFG optimization to evaluate the advantage of DFG optimization. The design constraints are 30ms, 30mJ, and $4\times10^9F^2$. The /L, /E, and /O labels, respectively, represent optimizations for latency, energy, and area. Benefiting from the compiler optimizations, each algorithm has varying degrees of improvement in terms of latency and energy consumption.

When the design objective is latency, compared with designs before optimization, MD5/L, RIPEMD-160/L, and SHA-1/L reduce the energy consumption by up to 5.7%, but the latency difference is not obvious. Similarly, when the design objective is energy, the energy differences of MD5/E, RIPEMD-160/E, and SHA-1/E are very small compared with designs before optimization while achieving a maximum

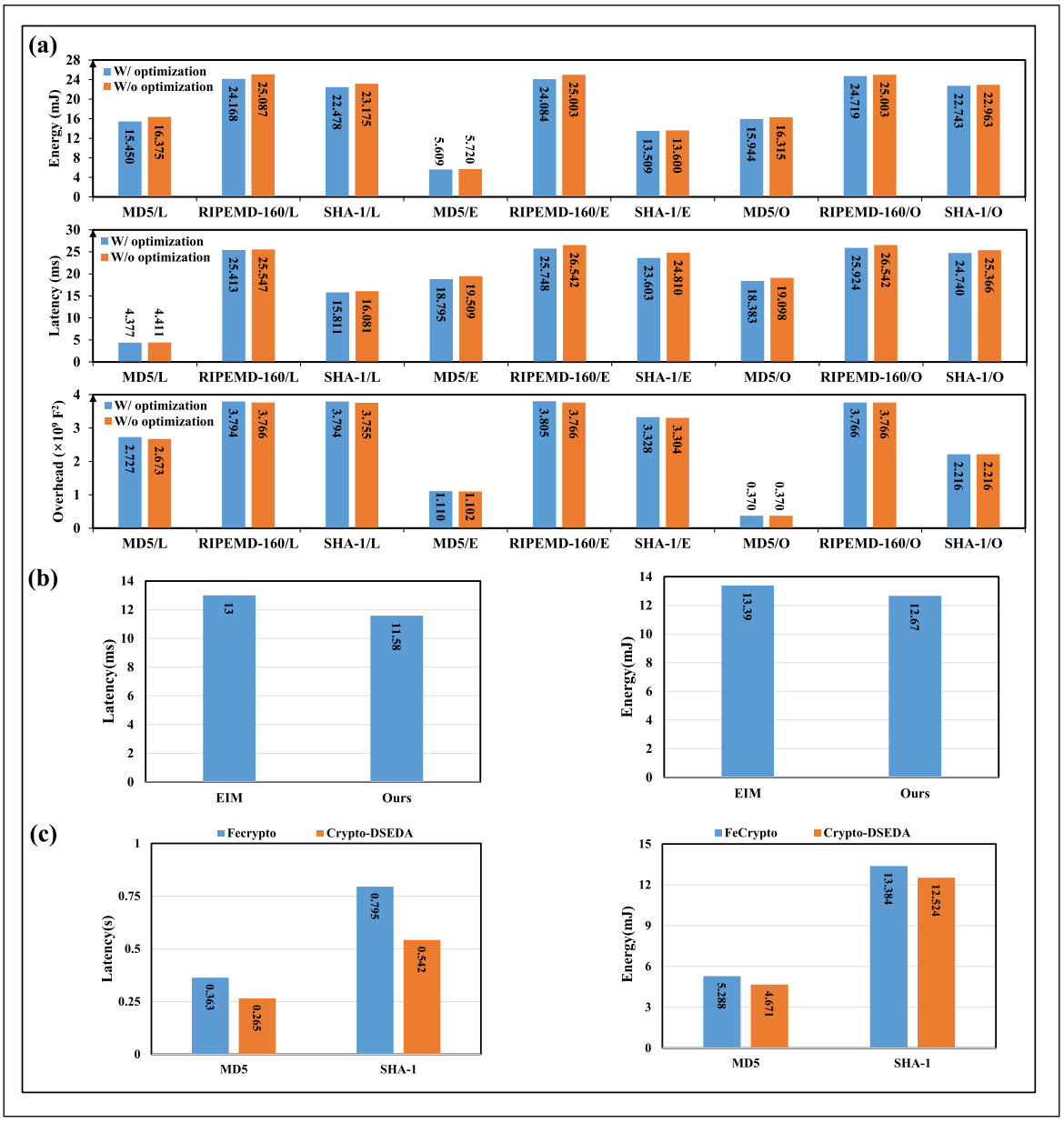


**Figure 3. (a) Energy, latency, and overhead comparisons for encrypting 1-GB data. (b) Latency and energy comparisons for encrypting 1-GB data with AES. (c) Energy and latency comparison for encrypting 1-GB data with MD5 and SHA-1.**

reduction of 4.9% in latency. The reason is that consecutive NOT and AND operations are detected and replaced with IMP operations to improve performance when the design objective is latency or energy. When performing latency/energy optimization, choosing low-latency/energy modules leads to insignificant differences in latency/energy before and after optimization, so more differences are manifested in the other aspect.

For area optimization, each design maintains the original data flow to avoid additional area overhead. However, by eliminating redundant operations and redundant memory accesses, the MD5 architecture can reduce 16 XOR operations, and the RIPEMD-16 architecture and the SHA-1 architecture can, respectively, reduce 32 and 20 read operations. The maximum differences in latency and energy consumption before and after optimization are 3.8% and 2.3%, respectively.

### Comparison with CiM-based accelerator

Two recent CiM-based accelerators, EIM [2] and FeCrypto [7], are used for comparison. We use Crypto-DSEDA with the same architecture configuration and device parameters to explore AES, MD5, and SHA-1 architectures. Latency and energy comparisons are shown in Figure 3b and c.

Compared to EIM, Crypto-DSEDA reduces the latency and energy by 10.9% and 5.7%, respectively. The main reason is that operator scheduling allows for the full utilization of the memory array, effectively reducing the stagnation of the CiM operation in the write-back and column mixing phases. In addition, redundant XOR operations are also eliminated.

Benefiting from iterative optimization and DFG optimization, Crypto-DSEDA achieves a maximum energy reduction of 11.7% compared to FeCrypto. In addition, different from the multiple-purpose FeCrypto, Crypto-DSEDA does not require the placement of LUTs, allowing for more CIM arrays to be configured within each CU. As a result, Crypto-DSEDA reduces latency by up to 31.8%.

**Domain-specific accelerators** have become a new development trend of chips. However, their designs lack dedicated EDA tools. Domain-specific EDA tools may be developed by exploring the common features in the input, operations, data flow, and so on of the application domain. Cryptographic algorithms contain a large number of Boolean logic operations, which are extremely suitable for CiM acceleration. Existing cryptographic accelerators rely on expensive manual design efforts due to the difficulty of generating CiM architectures with traditional EDA tools. Aiming at this problem, we propose an automatic generation framework based on CiM architecture to find the objective-optimal accelerator under the given PPA constraints. The evaluation shows that our autogenerated accelerators exhibit performance and energy advantages compared with state-of-the-art CiM-based accelerators. ■

## Acknowledgments

This work was supported in part by the National Key Research and Development Program of China under Grant 2018YFA0701500; in part by the Strategic Priority Research Program of CAS under Grant XDB44000000; in part by the National Natural Science Foundation of China under Grant 62122076, Grant 62025404, Grant 92164108, and Grant U23A20322; in part by the Key Research Program of Frontier Sciences, CAS, under Grant ZDBS-LY-JSC012; in part by the Innovation Funding of Institute of Computing Technology, CAS, under Grant E261040; in part by the Youth Innovation Promotion Association CAS; in part by the Hunan Provincial Natural Science Foundation under Grant 2023JJ50009 and Grant 2023JJ30599; and in part by the Post-graduate Scientific Research Innovation Project of Xiangtan University under Grant XDCX2023Y184.

## References

[1] M. Xie et al., "Securing emerging nonvolatile main memory with fast and energy-efficient AES in-memory implementation," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 26, no. 11, pp. 2443–2455, Nov. 2018.

[2] M. Ma et al., "Efficient in-memory AES encryption implementation using a general memristive logic: Surmounting the data movement bottleneck," *IEEE Nanotechnol. Mag.*, vol. 16, no. 2, pp. 24–C3, Apr. 2022.

[3] X. Chen, Y. Wu, and Y. Han, "FePIM: Contention-free in-memory computing based on ferroelectric field-effect transistors," in *Proc. 26th Asia South Pacific Design Autom. Conf. (ASP-DAC)*, Jan. 2021, pp. 114–119.

[4] A. Agrawal et al., "X-SRAM: Enabling in-memory Boolean computations in CMOS static random access memories," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 65, no. 12, pp. 4219–4232, Dec. 2018.

[5] S. Chhabra and Y. Solihin, "I-NVMM: A secure non-volatile main memory system with incremental encryption," in *Proc. 38th Annu. Int. Symp. Comput. Archit. (ISCA)*, Jun. 2011, pp. 177–188.

[6] D. Fan, S. Angizi, and Z. He, "In-memory computing with spintronic devices," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI (ISVLSI)*, Jul. 2017, pp. 683–688.

[7] R. Liu et al., "FeCrypto: Instruction set architecture for cryptographic algorithms based on FeFET-based in-memory computing," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 42, no. 9, pp. 2889–2902, Sep. 2023.

[8] W.-H. Chen et al., "A 16 Mb dual-mode ReRAM macro with sub-14 ns computing-in-memory and memory functions enabled by self-write termination scheme," in *IEDM Tech. Dig.*, Dec. 2017, pp. 2–28.

[9] M. Lee et al., "FeFET-based low-power bitwise logic-in-memory with direct write-back and data-adaptive dynamic sensing interface," in *Proc. ACM/IEEE Int. Symp. Low Power Electron. Design*, Aug. 2020, pp. 127–132.

[10] X. Zhang et al., "Re-FeMAT: A reconfigurable multifunctional FeFET-based memory architecture," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 41, no. 11, pp. 5071–5084, Nov. 2022.

[11] W. Zhao et al., "New generation of predictive technology model for sub 45 nm early design exploration," *IEEE Trans. Electron Devices*, vol. 53, no. 11, pp. 2816–2823, Nov. 2006.

[12] K. Ni et al., "A circuit compatible accurate compact model for ferroelectric-FETs," in *Proc. IEEE Symp. VLSI Technol.*, Jun. 2018, pp. 131–132.

**Rui Liu** is pursuing a PhD at Xiangtan University, Xiangtan 411105, China. He is also at the Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100083, China. His research interests include EDA and computing-in-memory architecture design. Liu has a BS and an MS from Xiangtan University, Xiangtan, China.

**Zerun Li** is pursuing a PhD at the Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100083, China. His current research interests include accelerating large-scale graph processing on emerging storage devices. Li has a BS in electronic engineering from Tsinghua University, Beijing.

**Xiaoyu Zhang** is pursuing a PhD at the University of Chinese Academy of Sciences, Beijing 101408, China. His research interests include electronic design automation and computing-in-memory architecture design. Zhang has a BE from the School of Computer Science and Technology, University of Science and Technology of China, Hefei, China.

**Wanqian Li** is pursuing a PhD at the University of Chinese Academy of Sciences, Beijing 101408, China. Her research interests include electronic design automation and computing-in-memory architecture design. Li has a BS in microelectronics science and engineering from Nankai University, Tianjin, China.

**Libo Shen** is pursuing an master's at the University of Chinese Academy of Sciences, Beijing 101408, China. His research interests include electronic design automation and computing-in-memory architecture design. Shen has a BS from the School of Information and Communication Engineering, Beijing University of Posts and Telecommunications, Beijing.

**Rui Tang** is pursuing a master's at the University of Science and Technology of China (USTC), Hefei 230052, China. He is also at the Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100083, China. His research interests include EDA and computing-in-memory architecture design. Tang has a BS from USTC.

**Zhejian Luo** is pursuing an MS at the School of Materials Science and Engineering, Xiangtan University, Xiangtan 411105, China. He is also at the Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100083, China. His research interests include computing-in-memory architecture design. Luo has a BE from Xiangtan University.

**Xiaoming Chen** is an associate professor at the Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100083, China, Beijing. His research interests include computer-aided design for integrated circuits and computing-in-memory architectures. Chen has a BS and PhD in electronic engineering from Tsinghua University, Beijing. He is a Member of IEEE.

**Yinhe Han** is a professor at ICT, CAS, Beijing 100083, China. His research interests include microprocessor design, integrated circuit design, and computer architecture. Han has an MS and a PhD in computer science from the Institute of Computing Technology (ICT), Chinese Academy of Sciences (CAS). He is a Member of IEEE.

**Minghua Tang** is at the School of Materials Science and Engineering, Xiangtan University, Xiangtan 411105, China. His research interests include neuromorphological devices for computing-in-memory applications and power devices. Tang has a BS in physics and a PhD in materials physics and chemistry from Xiangtan University. He is a Member of IEEE.

■ Direct questions and comments about this article to Xiaoming Chen, Institute of Computing Technology, Chinese Academy of Science, Beijing 100083, China; University of Chinese Academy of Sciences, Beijing 101408, China; chenxiaoming@ict.ac.cn; Minghua Tang, School of Materials Science and Engineering, Xiangtan University, Xiangtan 411105, China; tangminghua@xtu.edu.cn.