Automata Designs for Data Encryption with AES using the Micron Automata Processor

Angkul Kongmunvattana

School of Computer Science, Columbus State University, Columbus, GA, USA

Summary
Cybersecurity has become the most important issue in the current era of cyber warfare. Significant advantages can be obtained from using co-processing units when a cyberattack diminished the computing power of the main processor cores from carried out useful tasks. Automata Processor (AP) is a novel accelerator from Micron Technology, which is based on the non-von Neumann architecture and the processing in memory concept. A series of automata designs for implementing the Advanced Encryption Standard algorithm on the AP board is proposed, validated, compiled, and simulated using AP SDK and AP Workbench. The results demonstrated that the proposed automata designs utilized the available resources on the AP chip efficiently, yielding the maximum degree of concurrencies across all six ranks available on the Micron AP-D480 board. Thus, this paper serves as an exploratory guide to enhance cybersecurity operational capability by using the AP co-processing board.

Key words:
Advanced encryption standard, Automata processor, Cyber security, Data encryption.

1. Introduction

Server and storage systems on the cloud handle tremendously large amount of data with multiple nodes connected to each other via the Internet. Data encryption protocols play an important role in maintaining data security, integrity, and privacy by obfuscating original data contents. For this general purpose of data security, advanced encryption standard (AES) algorithm has been widely adopted. In most cases, the tasks of data encryption and decryption are carried out in software by the processor cores. When the systems are under attack or infected by malwares, the processing speed is diminished due to a lack of CPU cycles to carry out these important tasks. This is an unacceptable scenario in the context of cyber warfare, where the capability to maintain security and availability of data while under attack is essential [1].

Recently, Micron Technology proposed a novel accelerator based on the concept of processing in memory and the non-von Neumann architecture called the Automata Processor (AP) [2]. The processing capability of the Micron AP is based on its data processing rate as well as the design and programming of its state transition elements (STEs), counter elements, and Boolean elements. A few seminal studies and preliminary results have shown that the AP technology is capable of solving problems in Bioinformatics [3] and Data Mining [4]. To the best of our knowledge, this is the first instance of data security application using the Micron AP, which can be helpful to the cybersecurity community as this new technology becomes available.

In this paper, we propose a series of automata designs for implementing the AES algorithm on the Micron AP. The first automaton is designed to recognize an 8-bit block pattern in an input stream and to produce an output based on the substitution values given in the S-Box. The second and third automata designs recognize an 8-bit block pattern in an input stream and produce an output based on the multiplication operations with {02} and {03} in GF(2^8), respectively. The fourth automaton combines the outputs from multiplications in GF(2^8) using addition operations in GF(2^8) to complete the MixColumns() transformation. The compilation and simulation results demonstrated that the proposed automata designs utilized only 57% of the STEs and 62.5% of the Boolean elements available on each of the (six) AP ranks, allowing the 48-core AP-D480 board to process six input data stream concurrently.

The rest of this paper is organized as follows. A concise summary on the AES operations is provided in Section 2. An overview on the Micron AP is described in Section 3. The proposed automata designs are presented in Section 4. The results in terms of STE utilization and expected data processing rate are discussed in Section 5. Our findings are summarized in Section 6.

2. Advanced Encryption Standard

The National Institute of Standards and Technology (NIST) published advanced encryption standard (AES) in 2001 [5]. The AES is based on Rijndael algorithm, which is a symmetric block cipher. The AES adopted a data block size of 128 bits with the cipher keys of 128, 192, or 256 bits in length. This paper focuses on the automata designs for 128-bit encryption, but it can certainly be expanded to other key lengths. In general, AES repeats ten rounds of byte-level substitutions, row-wise left rotations at byte granularity, and column-wise matrix multiplications in

Manuscript received July 5, 2015
Manuscript revised July 20, 2015

GF(2^8). Multiplication and addition operations in GF(2^8) can be carried out with bit shifting and XOR operations. The goal of AES is to obfuscate the original input data contents from unauthorized accesses.

3. Automata Processor

The Micron Automata Processor (AP) is an accelerator that can be programmed to execute a large number of Finite State Machines (FSM) in parallel to identify patterns in data streams as well as to process them. There are three types of elements in the AP called (i) state transition elements (STE), (ii) counter elements, and (iii) Boolean elements. Each STE is designed to recognize an input data value, which can be any character classes over the 8-bit symbols. These STEs are reconfigurable and can be reprogrammed to recognize new input data values. An automaton design connects these STEs through transitional links, which are only activated and led to destination STEs when an input data value is recognized by the STEs. There are a few special types of STEs called starting and reporting STEs. A starting STE usually acts as an initial state of the FSM. Multiple starting states are allowed, which enables parallel execution of multiple FSMs. A starting STE is further classified into two subtypes namely all-input-start STE and start-of-data STE. All-input-start STE can be activated by any input data value whereas start-of-data STE can only be activated by a recognized input data value of that STE. A reporting STE is typically used as an acceptance state in the FSM. For example, FSM recognizing the word ANT appearing anywhere in an input stream is shown in Figure 1, whereas FSM recognizing only the word ANT appearing at the beginning of an input stream is shown in Figure 2.

Fig. 1 Automaton with all-input-start and reporting STEs.

Fig. 2 Automaton with start-of-data and reporting STEs.

Apart from the STE, counter elements can be used to report the number of times a particular input data value has been recognized. For example, FSM recognizing the word ANT and counting whether it appears more than twice in an input stream is shown in Figure 3.

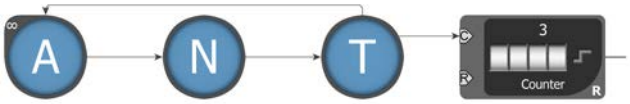

Fig. 3 Automaton with a counter element.

Finally, Boolean elements can be used to create a combinational circuit complementing the FSM designs. These Booleans elements can be programmed to act as AND, OR, NOT, NAND, NOR, AND-OR (i.e., sum-of-product), and OR-AND (i.e., product-of-sum) gates. An XOR gate used for an addition operation in GF(2^8) is implemented using AND, NAND, and OR gates as shown in Figure 4.

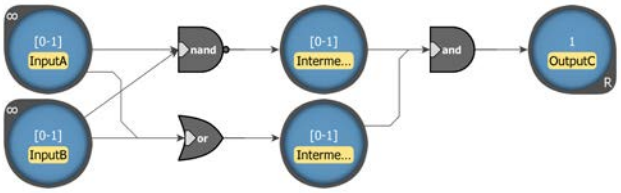

Fig. 4 Automaton with Boolean elements.

4. Automata Designs for AES

The first step in designing an automaton for advanced encryption standard (AES) is mapping its operations to pattern recognition problems. Then, one or more finite state machine (FSM) is developed to recognize the patterns and to produce desired outputs. Finally, the FSM is then programmed on to the STE, counter, and Boolean elements. Under the AES algorithm, the first common operation in multiple rounds of processing is substitution transformation using S-Box. The next common operation is multiplication in GF(2^8). Finally, the automaton for combining the multiplication outputs through addition operation in GF(2^8) is required. The explanation of each automaton is presented in the following subsections.

4.1 S-Box Substitution Transformation Automaton

Consider an automaton designed for an 8-bit sequence $B = b_1b_2b_3b_4b_5b_6b_7b_8$ that accepts any 8-bit binary strings and provides a mechanism to deduce an 8-bit output sequence based on the acceptance state reported. The proposed automaton design consists of two start-of-data STEs, $\sum_{i=2}^{8} 2^i$ regular STEs, and 2^8 reporting STEs arranged in the a binary tree fashion. Thus, this automaton requires 510 STEs. The two start-of-data STEs are the root nodes

and the reporting STEs are the leave nodes. The usage of start-of-data STEs allows the substitution transformation to occur at an 8-bit byte granularity in the input data stream. For example, when an 8-bit input data is 00000000_2, a substitution transformation using AES's S-Box produces 01100011_2 or 63_{16} as an output. Thus, this automaton is simply a one-to-one mapping between an 8-bit input data to the output data value provided in the S-Box. The design is best described through an illustration (see Figure 5).

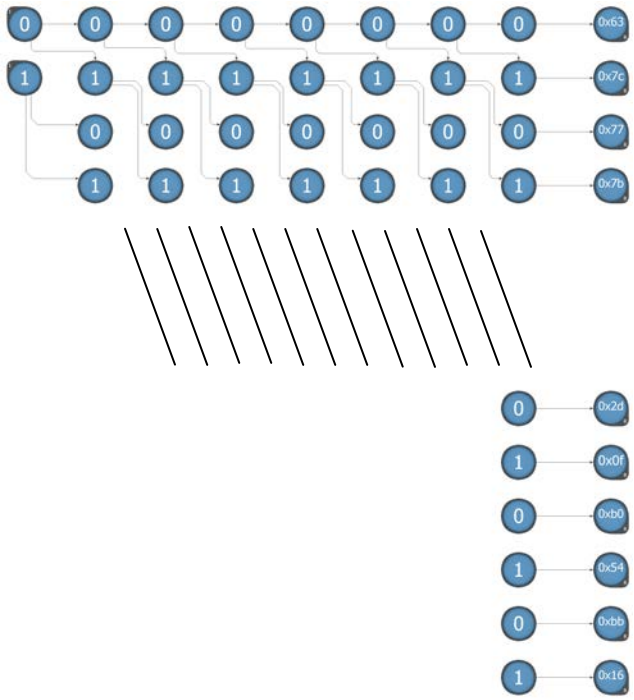

Fig. 5 Automaton for substitution transformation based on S-Box.

4.2 Automaton for Multiplication in GF(2^8)

For the MixColumn() transformation in AES, two multiplication operations in GF(2^8) are carried out on two bytes of data in each column. Specifically, the first multiplication operation is performed on an input byte and {02} in GF(2^8). An output from this multiplication operation is obtained by left shifting an input byte one bit, following by an exclusive-OR with {1b} when the left-most bit before the left shift was equal to "1". Thus, it is possible to pre-calculate the outputs of this multiplication in GF(2^8) for all possible 8-bit input sequences and to create an automaton that recognizes each of the 8-bit input data sequences with reporting STEs for deducing the outputs. For example, multiplying {00} with {02} produces {00} because left shifting 00000000_2 by one bit yields 00000000_2. Multiplying {01} with {02} produces {02} because left shifting 00000001_2 by one bit yields 00000010_2. Multiplying {02} with {02} produces {04} because left shifting 00000010_2 by one bit yields 00000100_2. Multiplying {03} with {02} produces {06} because left shifting 00000011_2 by one bit yields 00000110_2. These multiplication operations and their results are mapped to the first four rows of the automaton shown in Figure 6.

The last six rows of the automaton represents the multiplications of {02} with {fa}, {fb}, {fc}, {fd}, {fe}, and {ff}, respectively. Specifically, the bottom row of Figure 6 represents an output from recognizing an 8-bit input in the data stream being 11111111_2. Multiplying {ff} with {02} produces {e5} because left shifting 11111111_2 by one bit produces 11111110_2. Since the left-most bit of {ff} is 1, the output from bit shifting is then exclusive-ORed with {1b} (i.e., 11111110_2 XOR 00011011_2), which produces {e5} (i.e., 11100101_2). The next row up represents an output from recognizing an 8-bit input in the data stream being 11111110_2. Multiplying {fe} with {02} produces {e7} because left shifting 11111110_2 by one bit produces 11111100_2. Since the left-most bit of {fe} is 1, the output from bit shifting is then exclusive-ORed with {1b} (i.e., 11111100_2 XOR 00011011_2), which produces {e7} (i.e., 11100111_2). This automaton also requires 510 STEs because it is required to recognize all of the unique 8-bit input data values (i.e., 2^8 is 256). A depiction of the automaton is shown in Figure 6.

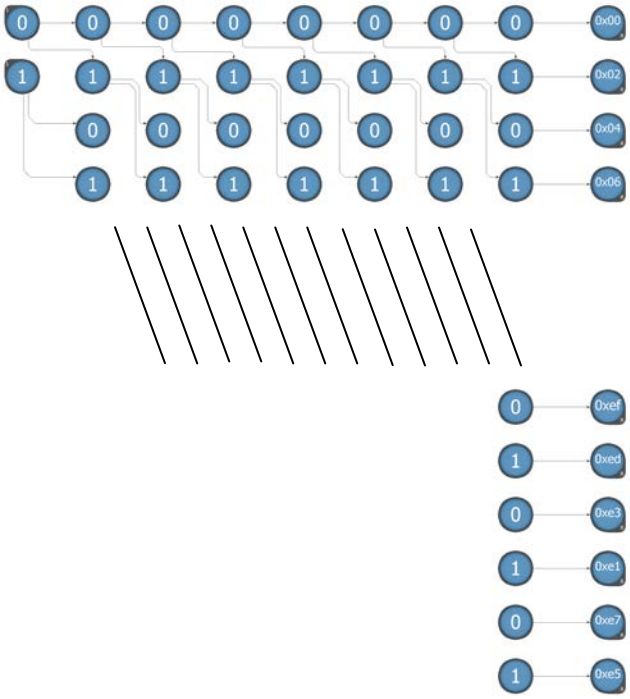

Fig. 6 Automaton for GF(2^8) multiplication of 8-bit input with {02}.

The second multiplication operation is performed on an input byte and {03} in GF(2^8). An output of this operation is obtained by performing multiplication with {02} in GF(2^8) as described in the last paragraph, and then, performing an addition operation in GF(2^8) on the output with the original 8-bit input. Again, it is possible to pre-calculate the outputs of this multiplication in GF(2^8) for all possible 8-bit input sequences and to create an automaton that recognizes each of the 8-bit input data sequences with reporting STEs for deducing the outputs.

For example, the 2nd row of an automaton shown in Figure 7 recognizes 00000001_2 (i.e., {01}) as an 8-bit input data. Multiplying {01} with {03} produces {03} because {01} multiplies by {02} produces {02} and {02} $\oplus$ {01} produces {03} (i.e., 00000010_2 XOR 00000001_2 = 00000011_2).

In another example, the bottom row of Figure 7 represents an output from recognizing an 8-bit input in the data stream being 11111111_2. Multiplying {ff} with {03} produces {1a} because left shifting 11111111_2 by one bit produces 11111110_2. Since the left-most bit of {ff} is 1, the output from bit shifting is then exclusive-ORed with {1b} (i.e., 11111110_2 XOR 00011011_2), which produces {e5} (i.e., 11100101_2). At this point, {e5} is an output from {ff} multiplied by {02}. To produce an output for {ff} multiplied by {03}, {e5} has to be added with {ff}. Addition operation in GF(2^8) is equivalent to a bit-wise exclusive-OR. Thus, {e5} $\oplus$ {ff} produces {1a} (i.e., 11100101_2 XOR 11111111_2 = 00011010_2), which is an output of multiplying {ff} with {03}. This automaton also requires 510 STEs.

4.3 Automaton for MixColumn() Transformation

Apart from the two multiplication operations in GF(2^8), the MixColumn() transformation requires an exclusive-OR (in lieu of addition operations in GF(2^8)) of all outputs from the multiplication operations. As shown earlier, an XOR gate is implemented by using AND, OR, and NAND gates on the Micron AP since its Boolean element does not support an XOR gate.

5. Results and Discussion

The performance of the proposed automata designs is evaluated using the AP SDK and Workbench version 1.4-11 from Micron Technology, assuming a 48-core AP-D480 board. Specifically, each AP board has 6 ranks, each with 8 processor cores (as shown in Figure 8). Each core occupies one AP chip, which is divided into two half-cores. Each half-core holds 96 blocks. Each block has 256 STEs, 4 counter elements, and 12 Boolean elements. Thus, a 48-core AP board has 2,359,296 STEs, 36,864 counters, and 110,592 Boolean elements. In terms of input streaming, each rank on the AP board can handle one input stream at the rate of 1 Gbps. Thus, a cumulative data processing rate for the AP board is 6 Gbps.

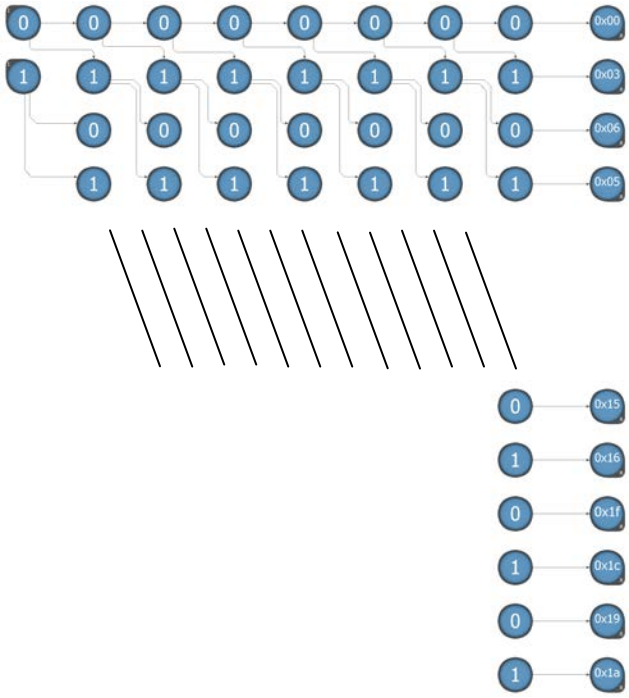

Fig. 7 Automaton for GF(2^8) multiplication of 8-bit input with {03}.

The proposed automata designs were successfully validated, compiled, and simulated on the AP Workbench. The substitution transformation using S-Box in each of the ten rounds of AES needs a total of 8,160 STEs. The MixColumn() transformation using multiplication and addition operations in GF(2^8) in each of the ten rounds of AES needs a total of 16,320 STEs and 1,152 Boolean elements. Thus, for ten rounds of AES, the proposed automata designs need a total of 224,800 STEs and 11,520 Boolean elements. Since each rank on the AP can hold 393,216 STEs and 18,432 Boolean elements, the proposed designs fit comfortable in one rank using only 57% of the STEs and 62.5% of the Boolean elements. Thus, six input data streams can be encrypted concurrently and independently on six ranks of the AP board.

Fig. 8 Eight Micron AP chips forming one rank.

In order to compare the performance of the proposed automata designs on the Micron AP board to prior work that used field programmable gate array (FPGA) devices or software running on the main CPU cores or the GPUs, a normalization of design and implementation lead time as well as technology is necessary. This is because designing, implementing, and optimizing pipelined AES circuits on the FPGA devices requires significantly more time and effort than designing and compiling automata for the Micron AP. Furthermore, the AP chip used 45nm technology whereas the current generation of CPUs and GPUs used 14nm and 28nm technology, respectively [6], [7]. To compare different architectures fairly, an assumption on linear scaling of clock frequency and square scaling of capacity to bring all architectures to the same semiconductor technology level can be made [8].

Table 1: Performance Comparison

Architectures	***Processing Rate***	***Throughput***
AP	133 MHz	6 Gbps [2]
Intel AES-NI	3.47 GHz	0.5 - 0.7 Gbps [9]
Nvidia GPU	900 MHz	14.6 Gbps [10]

Table 1 displays a collection of data on software implementation of AES running on Intel CPU cores and Nvidia GPUs. Intel AES-NI is a special class of assembly instructions tailored and optimized for software implementation of AES algorithm on Intel CPU cores. The AP implementation of AES significantly outperforms Intel AES-NI. Using linear scaling of clock frequency based on the semiconductor technology used to fabricate the AP chip and the GPU (i.e., 45nm vs 28nm), the software implementation of AES running on the GPU slightly outperforms the AES implementation on the AP.

5. Conclusion

The Micron Automata Processor is a new accelerator that is capable of processing multiple input data streams concurrently at a high rate. This paper explores its usage in the area of data security to aid the development of resilient computer systems that can withstand cyberattacks and also maintain a high degree of cybersecurity operational capability. A series of automata designs for implementing advanced encryption standard algorithm was proposed, validated, compiled, and simulated. The performance results demonstrated a promising future of the Automata Processor as an accelerator in the area of cybersecurity.

Acknowledgments

The author thanks Micron Technology, Inc. for granting a limited license and an access to a pre-release version of the AP SDK and AP Workbench. This research is supported in part by computing resources from the NSF XSEDE Startup Allocation Award.

References

[1] D. Parr, "Securing the Cloud," *Journal of Information Warfare*, 13(2):56-69, April 2014.
[2] P. Dlugosch, D. Brown, P. Glendenning, M. Leventhal, and H. Noyes, "An Efficient and Scalable Semiconductor Architecture for Parallel Automata Processing," *IEEE Transactions on Parallel and Distributed Systems*, 25(12):3088-3098, December 2014.
[3] I. Roy and S. Aluru, "Discovering Motifs in Biological Sequences using the Micron Automata Processor," *IEEE Transactions on Computational Biology and Bioinformatics*, 2015, in press.
[4] K. Wang, M. Stan, and K. Skadron, "Association Rule Mining with the Micron Automata Processor," In *Proceedings of the 29th IEEE International Parallel and Distributed Processing Symposium*, 2015, in press.
[5] National Institute of Standards and Technology (NIST), "Advanced Encryption Standard (AES)," FIPS Publication 197, November 2001.
[6] B. Dally, "GPU Computing: To ExaScale and Beyond," Plenary Speaker, NVIDIA Showcase, *The 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis (SC10)*, November 2010.
[7] Intel Corporation, "Advancing Moore's Law – The Road to 14 nm," Intel White Paper, August 2014.
[8] J. M. Rabaey, A. Chandrakasan, and B. Nikolic, Digital Integrated Circuits: A Design Perspective, 2nd Edition, Prentice Hall, January 2003.
[9] A. Basu and A. Bhargav-Santzel, "Intel AES-NI Performance Testing on Linux/Java Stack," Intel White Paper, June 2012.
[10] J. W. Bos, D. A. Osvik, and D. Stefan, "Fast Implementations of AES on Various Platforms," *Software Performance Enhancement for Encryption and Decryption and Cryptographic Compilers (SPEED-CC)*, October 2009.

Angkul Kongmunvattana earned a doctorate degree in Computer Science from the University of Louisiana at Lafayette in 1999. He is currently an Associate Professor in the School of Computer Science at Columbus State University. His area of research includes bioinformatics, cybersecurity, and high-performance computing systems. His research work has been supported in part by US National Science Foundation, US Geological Survey, Altera Corporation, and Sun Microsystems (now Oracle).