Analysis of AES Hardware Implementations

Song J. Park
Department of Electrical & Computer Engineering
Oregon State University, Corvallis, Oregon 97331
E-mail: parkso@ece.orst.edu

Abstract— **Following paper examines hardware implementation methods regarding Advanced Encryption Standard (AES). Compared to software implementation, migrating to hardware provides higher level of security and faster encryption speed. An overview of existing AES hardware implementation techniques are summarized. Then the direction of reconfigurable coprocessor as a cryptography hardware is proposed.**

Keywords— **AES, Rijndael, coprocessor, hardware implementation, optimization.**

I. Introduction

In October of 2000, the National Institute of Standards and Technology (NIST) announced Rijndael as the winner of Advanced Encryption Standard (AES) contest, in effort to address threatened key size of Data Encryption Standard (DES). Currently, most AES algorithm are implemented in software [1]. In software approach, the secret key is vulnerable to attacks and rely on underlying programs [2]. In addition, achievable speed by software implementation is not acceptable for internet applications such as routers. This leads to hardware design of AES where parallel processing and pipelining is possible [3]. Therefore, hardware systems offer superior performance with higher throughput. Helion Technology claims that speed exceeding 16 Gbps for FPGA and 25 Gbps for ASIC design is available [4]. Moreover, hardware implementations are considered physically secure since tempering by an outside attacker is difficult.

II. Rijndael Algorithm

A brief introduction of Rijndael encryption and decryption algorithm is provided. Rijndael is a block cipher developed by Joan Daemen and Vincent Rijmen. The algorithm is flexible in that any combination of data and key size of 128, 192, and 256 bits are supported. However, AES only allows the data length to be 128 bits while conserving the property of supporting three different key lengths. AES can be divided into four basic operation blocks which operates on array of bytes, organized as a 4×4 matrix called the state. Four basic steps, called layers consist of the ByteSub Transformation, the ShiftRow Transformation, the MixColumn Transformation, and AddRoundKey [5].

• The ByteSub Transformation:
Non-linear byte substitution which is composed of multiplicative inverse and affine transformation.

Author is a graduate student at the Department of Electrical & Computer Engineering, Oregon State University, Corvallis, Oregon 97331. E-mail: parkso@ece.orst.edu

• The ShiftRow Transformation:
Linear diffusion process, operating on individual rows. Depending on the row location, offset of left shift varies from zero to three bytes.

• The MixColumn Transformation:
Matrix multiplication over $GF(2^8)$. Column vector is multiplied with a fixed matrix where the bytes are treated as a polynomials rather than numbers.

• AddRoundKey:
Simple byte XOR operation with the round key.

These four layer steps describe one round of AES. A 128 bit round key, used in AddRoundKey operation, is generated by the key schedule. Sub-keys are derived from the original user key by XOR operation of two previous columns. For columns that are in multiples of four, the process involves additional round constants, S-box, and shift operation.

Excluding the first and the last round, AES encryption round executes nine iterations. First round of the encryption step performs XOR with the original key and the last round skipss MixColumn layer.

All four layers described above have corresponding inverse operations such that the decryption is simply the reverse order operations of these inverse transformations. Note that the constant matrix for the MixColumn multiplication used in the decryption process consist of higher values. The result is more complex decryption unit compared to the hardware for encryption.

III. Hardware Implementations

Currently, several hardware implementation methods have been designed and published. There are many design choices encountered during hardware implementation of AES. In reality, these choices will be limited to its applications and budget. From the perspective of performance, major decision lies in the tradeoff between area and speed. For example, fast system is obtained at a cost of increased area, and vice versa.

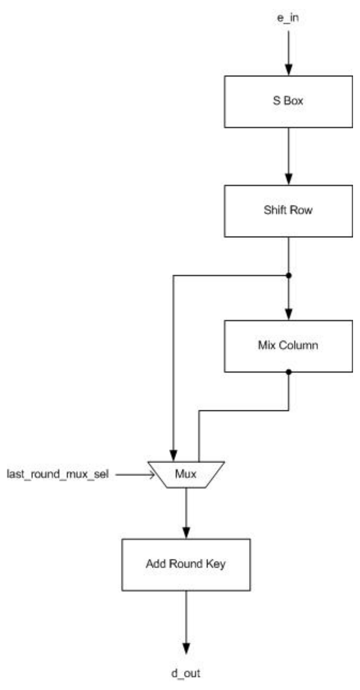
Before looking into different hardware architectures, basic hardware concepts are defined.

• Pipelining: Replicating rounds and placing registers in between. Increases throughput.
• Iterative Looping: One round of hardware design, which forces teh algorithm to reuse the same hardware.
• Loop unrolling: Refers to the process of unrolling multiple rounds.

• Latency: An elapsed time between start to finish of encryption.

For modes with feedback operations, pipelined design has no additional advantage since the encryption depends on the previous results. Otherwise, pipelining can increase the bandwidth, although area would be increased. For area sensitive applications, iterative looping design offers small area running at slower speeds.

Figure 1: Iterative Loop Design.
S-box consumes 84% of area.



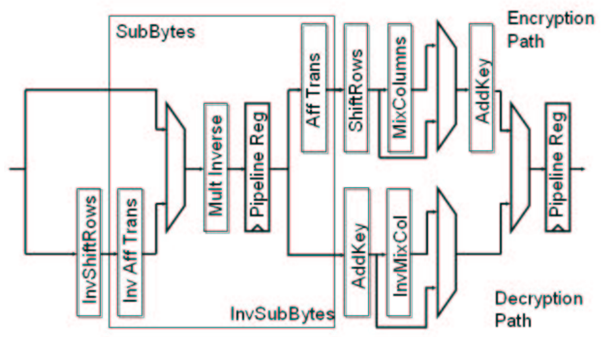
A. S-Box Optimization

In AES hardware implementation, S-box design contributes a major role in optimization. There are two approaches for S-box design:
1. Design a multiplicative inversion and affine transformation seperately
2. Construct a logic circuit defining the input and output of the S-box function
For a decomposed design, consisting of multiplicative inversion and affine transformation, area minimizations are possible using mathematical theorems. Previous research in this area include compact inversion circuits over $GF(2^8)$ based on Fermat's Little Theorem, low latency algorithm, and extended Euclid's algorithm [6]. Second method allows faster implementation where EDA synthesis tools create an optimized S-box structure.

An iterative loop based, AES encryption unit was designed in VHDL (VHSIC Hardware Description Language). According to the systhesis result, area of a S-box respect to all the hardware in one round is 84.4% [7]. Note that the result only relates to the encryption unit. A novel idea to minimize S-box space is to share the encryption and decryption Look-up Tables (LUT). The idea is to have one set of encryption and one set of decryption S-box unit in ROM. Then, S-boxes are reused for both encryption and decryption by initializing the LUTs corresponding to the current operation [8]. This method only requires S-box area that is equivalent to one encryption unit.

With Similar purpose, [9] proposes a modified decryption algorithm such that the order of inverse operation is equivalent to the order of encryption. Although, significance of the operations for decryption is different, this modification allows optimal pipelining. In addition, the SubByte operation shares the multiplicative inverse step for both encryption and decryption as shown in Fig. 2 [9]. As mentioned before, effectively utilizing the SubByte operation saves substantial amount of area.

Figure 2: Pipelined Architecture.



B. Key Scheduling

Two options exist for designing key schedule function. First method is to generate all the required sub-keys in advance or in beginning. This design requires a huge buffer for storing all the expanded keys. Second option is to generate sub-key in parallel with the encryption round. Second method requires much less buffer space, since only one 128 bit sub-key is produced and stored. Plus, the cold start latency of sub-key generation is smaller for this method. Unfortunately, the decryption unit uses the sub-keys in reverse order, forcing pre-generation of all sub-keys.

C. Cryptanalysis

An important consideration for any cryptographic system is the possible attacks against the algorithm. Side-channel attacks refers to a cryptanalysis method where the secret key information is attacked by gathering certain physical measurements on the electronic device [10]. Measurements include power consumption, the time, and electromagnetic radiation. In particular, power analysis attacks are powerful as they do not require expensive mathmetical equipment nor deep understanding of the algorithm. Basic concept behind power analysis attack is that statistical samples of the given key and output results are stored in beforehand. Then the retrieved collection of data is used to assist in recovering the secret key. Therefore, countermeasures should be imposed and applied, in order to be resistant to these attacks. However, the countermeasures against these attacks are typically costly to the speed and to the area of the system [10].

Strategies to combat side-channel attacks concentrate on de-correlating the output and randomization. For instance, introducing random timing shifts, adding dummy instructions, and applying random mask are techniques that attemp to counteract the side-channel attacks.

D. Routing

Although efficiency of hardware implementation was one of the evaluation criteria for choosing AES, only few hardware designs are presented for FPGA or ASIC platforms [9]. Analysis of routing was not mentions in most papers. For FPGA target, routing placement is predetermined within the FPGA architecture and this is the cause for greater area in FPGAs compared to ASIC designs. On contrary, area of ASIC designs would be greatly affected by routing overhead, where the minimum bus length is 128 bit. ASIC's floorplan by [9] report that the area estimations of routing were off by a factor of two in Synopsis. Previous experience with layout CAD tools helped me to realize the complexity of routing problems and its effect on chip area. In order to achieve optimized ASIC hardware design of AES, efficient routing algorithm is mandatory. Especially, with heightened undesirable properties of wires, minimizing wire space is equally important as optimizing core functional blocks.

Characteristics of ASIC, FPGA, and software implementations are summarized in a table [3].

Table 1: Implementation Comparison.

	ASIC	**FPGA**	**Software**
Parallel Processing	yes	tes	limited
Pipelining	yes	yes	limited
Speed	very fast	fast	moderate
Temper Resistance	strong	strong	weak
Design Cycle	long	moderate	short

IV. FPGA Coprocessor

Thus far, sections examined a stand alone AES hardware implementations. Following section describes the design which supports a coprocessor. Organization of this architecture consist of a CPU with an aid of a FPGA coprocessor. Coprocessor design integrates software and hardware into a single system, along with the reconfigurable capability of FPGA. Generally, the CPU controls the overall system operations while FPGA is responsible for calculations involving extensive computations. Moreover, FPGA is reconfigurable that can be reprogrammed in few milliseconds. As an example, consider an embedded system with a coprocessor, connected over a network using an ethernet. Depending on different situations, the FPGA can be reprogrammed dynamically according to the real-time status [11]. This allows FPGA to dynamically adjust to satisfy its surrounding requests.

A. DES Experiment

Hardware platform that was researched is an embedded system by Wind River. The physical setup of this board include an IBM PowerPC and Xilinx FPGA daughter card connected through a custom peripheral bus [12]. As a real world experiment, triple DES cipher was implemented on FPGA coprocessor. The result showed that an impressive 13 times speed improvement over algorithm running on software [13]. This experiment was an initial phase of CPU and FPGA construction, where large portion of DES functionality resided in the FPGA.

B. Issues

Many issues arise for the mixed system. First, communication between the processor and the FPGA must be managed. Type of communication protocol, management of control signals, and handling of crossing data needs to be developed. Second, debugging and simulating the coprocessor design should support the combination of software and hardware. Lastly, limited speed of the bus, connecting CPU and FPGA should be efficiently utilized. Designer should avoid degrading the overall system's performance by analyzing these parameters. Speed of the bus is the major bottleneck imposed on coprocessor systems. This is similar to the memory gap between processor and memory in the PC industry. I assume that coping with this problem of bus speed would be a major concern for the future.

C. Partitioning

Partitioning of the encryption algorithm between software and hardware is an interesting topic for research. During a partition process, a designer should account for factors relating to bus speed and inner communication. For dividing the AES system, one idea would be to allow the processor to compute shift operation and assign rest to the FPGA.

- CPU - ShiftRow and Control
- FPGA - SubByte, MixColumn, and KeyAddition

Shift operations in hardware represent mere wiring. Moving this function may benefit the hardware implementation due to reduced amount of wire interconnects. In other words, decreased wire parasitics and smaller routing overhead. Moreover, most of the AES function would operate on the FPGA and thus minimal communication occurs between the CPU and the FPGA.

V. Future Work

A major bottleneck with the coprocessor system is the slow communication link between the CPU and the FPGA. More research in this area is required to determine a way to effectively utilize the FPGA coprocessor that is connected through a slower bus. Additionally, concepts relating to decomposing an algorithm for software and hardware requires further research. Because coprocessor system combines software and hardware, development of simulators and design environment which supports both software and hardware is needed.

VI. Conclusion

An analysis of AES hardware implementations were examined in this paper. By implementing AES algorithm on a hardware, greater throughput and higher security is achieved. Major optimization methods were concentrated on the S-box design, which consumed most of layout area. The concept of mixed software and hardware design generates a new area of interest, although there are many unanswered questions.

References

[1] A. Elbirt, *Reconfigurable Computing for Symmetric-Key Algorithms*, Ph.D. thesis, Department of Electrical Engineering, Worcester Polytechnic Institute, 2002.

[2] P. Gutmann, “An open-source cryptographic coprocessor,” .

[3] K. Gaj and P. Chodowiec, “Hardware performance of the AES finalists-Survey and Analysis of results,” .

[4] “Helion Technology,” Tech. Rep., Helion, 2003.

[5] W. Trappe and L. Washington, *Introduction to Cryptography with Coding Theory*, Prentice Hall, New Jersey, 2002.

[6] S. Morioka and A. Satoh, “An optimized s-box circuit architecture for low power aes design,” in *Cryptographic Hardware and Embedded Systems - CHES 2002*, Ç. K. Koç and C. Paar, Eds. Aug 13–15, 2002, Forth International Workshop, Redwood Shores, USA, pp. 172–186, Springer-Verlag.

[7] B. Megarajan and S. Park, “Hardware implementation of aes (rijndael),” 2002, Webpage.

[8] J.V. McCanny M. McLoone, “High performance single-chip fpga rijndael algorithm implementation,” in *Cryptographic Hardware and Embedded Systems - CHES 2001*, Ç. K. Koç and C. Paar, Eds. May 14–16, 2001, Third International Workshop, Paris, France, pp. 65–76, Springer-Verlag.

[9] A. Lutz, J. Treichler, F. Frkaynak, H. Kaeslin, G. Basler, A. Erni, S. Reichmuth, P. Rommens, S. Oetiker, and W. Fichtner, “2gbit/s hardware realizations of rijndael and serpent: A comparative analysis,” in *Cryptographic Hardware and Embedded Systems - CHES 2002*, Ç. K. Koç and C. Paar, Eds. Aug 13–15, 2002, Forth International Workshop, Redwood Shores, USA, pp. 144–158, Springer-Verlag.

[10] J. Golic and C. Tymen, “Multiplicative masking and power analysis of aes,” in *Cryptographic Hardware and Embedded Systems - CHES 2002*, Ç. K. Koç and C. Paar, Eds. Aug 13–15, 2002, Forth International Workshop, Redwood Shores, USA, pp. 198–212, Springer-Verlag.

[11] “Re-Configurable Computing,” Tech. Rep., Wind River, August 2002, White Paper.

[12] “Hardware Reference Designs for SBC405GP,” Tech. Rep., Wind River, 2001.

[13] “Real World Experiences Designing For Mixed CPU + FPGA Systems,” Tech. Rep., Celoxica, August 2002, White Paper.