Cryptographic Coprocessor Design for IoT Sensor Nodes

Weizhen Wang, Jun Han*, Zhicheng Xie, Shan Huang and Xiaoyang Zeng
State Key Laboratory of ASIC and System, Fudan University
Shanghai, China
*Email: junhan@fudan.edu.cn

Abstract—**Together with the popularity of internet of things (IoT), the information security in IoT is becoming an urgent issue. In this paper, a cryptographic coprocessor is designed to provide security for IoT sensor nodes. This design incorporates the application-specific instruction set to support popular cryptography algorithms like advanced encryption standard (AES) and elliptic curve cryptography (ECC). The tailored instruction provides good flexibility, high energy efficiency and low latency. Local instruction and data ram is integrated into our coprocessor to reduce the instruction transfer between embedded processor and the coprocessor. Our work was synthesized under TSMC 65 nm technology. The measurement results show that our design consumes 32.7 uJ for each ECC point multiplication and 3 nJ for each AES encryption and decryption when working at 10 MHz.**

Keywords: IoT, Cryptography, ECC, AES, Application-specific instruction set processor (ASIP)

I. Introduction

The advancement in IoT is not only making our life more convenient but also leading to some serious problems. Amongst them, the potential risk of sensitive data leakage is one of the most concerned problems. The conventional solution is to utilize the public-key cryptography (PKC) for secure key exchange along with the symmetric-key cryptography (SKC) like AES for secure data transfer. However, due to the strict power and resources constraints in IoT sensor nodes, software implementation of encryption algorithms with high efficiency is proved to be hard. The SoC implementation with hardware accelerator or dedicated hardware for both PKC and SKC has gained a lot of attention for the reason of energy efficiency and acceptable performance.

Since modular multiplication is the kernel operation in PKC and consumes a lot of time and energy, various kinds of modular multipliers have been proposed and integrated into SoC as accelerators to reduce the ECC runtime. However, previous works show that the approach of integrating dedicated hardware into SoC results in massive data and instructions transferred between the embedded processor and the dedicated hardware [1] and becomes the performance bottleneck of the whole system. In this paper, we propose and implement a cryptographic ASIP with local instruction and data storage to speed up the cryptographic tasks with little data and instruction transfer.

ECC and RSA are the two popular PKC algorithms. Both of them can be used to provide efficient solution for authentication in communication systems. ECC compared with RSA uses much smaller key size when provides the same level of security [2]. What's more, the implementation of ECC in F_{2^m} consumes less power when when compared with that in F_P with the same key size [3]. Thus the ECC over F_{2^m} is an proper choice for applications like IoT sensor nodes where power and resources are limited.

Since AES is one of the most popular encryption algorithm for data transfer, instructions for AES key expansion, encryption and decryption are also included in our customized instruction set. As a result, our coprocessor could be programmed to support AES encryption and decryption with different operation modes like cipher block chaining (CBC), electronic code book (ECB) and offset code book (OCB).

This paper is organized as follows. In section II, we introduce the implementation of the proposed cryptographic coprocessor. In section III, the SoC platform which our coprocessor is integrated with is introduced. Moreover, the performance evaluation of the proposed coprocessor is also provided in this section. Finally, the conclusion is obtained in section VI.

II. Cryptographic Coprocessor Architecture

The overall architecture of the coprocessor with a 5-stage pipeline is illustrated in Fig. 1.

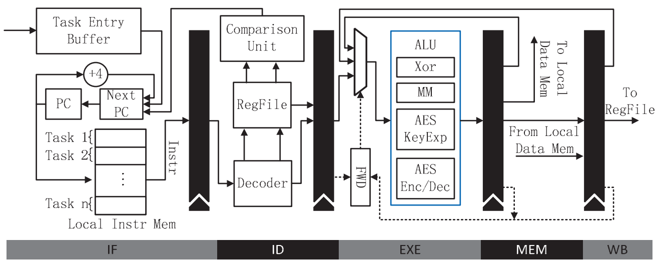

Figure 1. Hardware Architecture for the Cryptographic Coprocessor

Several tasks like point doubling, point addition, point multiplication, AES key expansion, AES encryption and decryption are stored in the local instruction memory. Entries of those tasks are stored in a lookup table, namely, the task

This work was supported by the National Natural Science Foundation of China (61574040, 61234002, 61525401), and the Project of the State Key Laboratory of ASIC and System (2015ZD003).

entry buffer in Fig. 1. When the coprocessor start working, the program counter (PC) is set to the entry of a certain task so as to control coprocessor to perform the required task. Since the instruction memory is pre-programmed, the coprocessor is able to work continually.

The FWD unit in Fig. 1 is a forwarding control unit, which is utilized to resolve the data hazard and improve the throughput. The arithmetic logic unit (ALU) mainly consists of a Montgomery modular multiplier (MM), XOR, AES key expansion and AES encryption and decryption unit. The digit serial Montgomery modular multiplier over F_2^m proposed in [4] is implemented with digit size w=16. Moreover, instead of using irreducible polynomial with fixed degree, our design could support irreducible polynomial with different degrees. The maximum degree supported is 256. The irreducible polynomials $P(x)=x^{163}+x^7+x^6+x^3+1$ and $P(x)=x^{233}+x^{74}+1$ ecommended by National Institute of Standards and Technology (NIST) are supported by our design. The implementation of Sbox transformations with lookup table results in high throughput but consumes a lot of area, which is not capable for applications like IoT sensor nodes. In our design, the Sbox transformations are implemented based on the composite-field to reduce the area consumption [5]. The register file is used to store the subkeys generated by key expansion unit, and the generated subkeys are used by AES encryption and decryption unit, which fulfills one AES round.

III. SoC Verification and Performance Evaluation

The proposed coprocessor is integrated with SoC platform depicted in Fig. 2. The DataIn, DataOut and TaskReg registers are mapped to memory address values so that they could be accessed by the embedded processor in the same way that it accesses memory. The TaskReg registers receive commands, named macro instructions, from the embedded processor and the macro instructions are decoded to index the task entry in task entry buffer shown in Fig. 1. Since the local data and instruction memory is deployed in our coprocessor, only indispensable data is transferred between embedded processor (OpenRISC 1200) and the coprocessor.

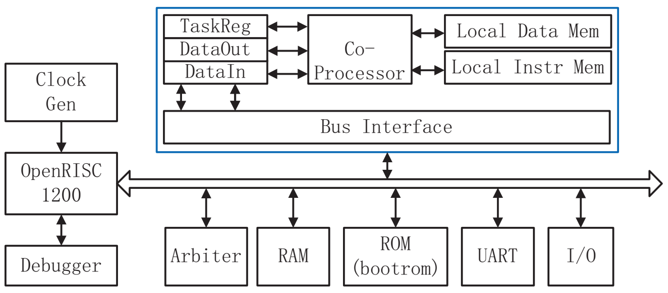

Figure 2. SoC Verification

Our work was synthesized under TSMC 65 nm technology. The synthesis results show that the coprocessor requires 162 kGates and 36% of the total area is consumed by the local storage. When it comes to the runtime of each ECC point multiplication, compared with previous works listed in Table I, our work outperforms previous works with speedup of 34, 1.61 and 1.40, respectively. It should be noted that our design could be configured for irreducible polynomials with different degrees and the supported key size is larger than previous works, which means our design offers security of higher level.

Table I. Runtime for Each Point Multiplication

Works	Key Size	Cycles	Comment
Wenger *et al.* [6]	163	7216905	Software, openMSP 430
Wenger *et al.* [6][a]	163	341835	Dedicated hardware
Hein *et al.* [7]	163	296299	Dedicated hardware
This Work	233	212270	ASIP

a. The result without squarer unit is selected for a fair comparison

Table II. Energy Consumed by the Coprocessor[a]

Task	Cycles	Power (mW)	Energy (nJ)
AES key expansion	12	2.11	2.53
AES encryption	14	2.13	2.98
AES decryption	14	2.23	3.12
ECC point multiplication[b]	212270	1.54	32689.58

a. Clocked @ 10MHz, Power supply @ 1.2V

b. The NIST recommended curve B-233 [8] is selected for measurement

The energy consumption when the proposed coprocessor performs certain tasks is measured with Synopsys PrimeTime PX and the results are shown in Table II. As shown in Table II, the coprocessor consumes approximately 3 nJ for each AES encryption and decryption

IV. Conclusion

In this paper, a cryptographic coprocessor for IoT applications is proposed. The coprocessor incorporates local data and instruction memory to save the bandwidth when it is integrated with SoC. The tailored instruction was capable of supporting ECC over F_2^m and AES algorithms. The simulation results show that our design achieves small latency compared with related works. The power estimation results show good energy efficiency for cryptographic tasks.

[1] Xu Guo and Patrick Schaumont. Optimizing the hw/sw boundary of an ecc soc design using control hierarchy and distributed storage. In Proceedings of the Conference on Design, Automation and Test in Europe, pages 454–459. European Design and Automation Association, 2009.

[2] Neal Koblitz. Towards a quarter-century of public key cryptography. Springer, 2000.

[3] Erich Wenger and Michael Hutter. Exploring the design space of prime field vs. binary field ecc-hardware implementations. In Nordic Conference on Secure IT Systems, pages 256–271. Springer, 2011.

[4] Miroslav Knezevic, Frederik Vercauteren, and Ingrid Verbauwhede. Speeding up barrett and montgomery modular multiplications.

[5] Atri Rudra, Pradeep K Dubey, Charanjit S Jutla, Vijay Kumar, Josyula R Rao, and Pankaj Rohatgi. Efficient rijndael encryption implementation with composite field arithmetic. In International Workshop on Cryptographic Hardware and Embedded Systems, pages 171–184. Springer, 2001.

[6] Erich Wenger. Hardware architectures for msp430-based wireless sensor nodes performing elliptic curve cryptography. In International Conference on Applied Cryptography and Network Security, pages 290–306. Springer, 2013.

[7] Daniel Hein, Johannes Wolkerstorfer, and Norbert Felber. Ecc is ready for rfid–a proof in silicon. In International Workshop on Selected Areas in Cryptography, pages 401–413. Springer, 2008.

[8] PUB FIPS. 186-4. Digital Signature Standard (DSS), 2013