Available online at www.elixirpublishers.com (Elixir International Journal)

Electrical Engineering

Elixir Elec. Engg. 74 (2014) 26633-26636

ISSN: 2229-712X

Simulation & Synthesis of a Cryptography Processor for Portable Electronic Devices

M.Aravind Kumar, A.Krishna Chaitanya Varma and P.S.Maitrey
Department of ECE, Vishnu Institute of Technology, Bhimavaram, AP, India.

ARTICLE INFO

Article history:
Received: 22 June 2013;
Received in revised form:
20 August 2014;
Accepted: 29 August 2014;

Keywords
Synthesis,
Cryptography processor,
AES, DES,
ECC.

ABSTRACT
Cryptography circuits for portable electronic devices provide user authentication and secure data communication. These circuits should, in general, occupy small chip area, consume low power, handle several cryptography algorithms, and provide acceptable performance. This paper presents the simulation and synthesis of three standard cryptography algorithms on a universal architecture. The cryptography processor implements both private key and public key algorithms and meets the power and performance specifications. The mentor graphics modelsim tool is used for design and simulation and also Synopsys Design Compiler tool is used for synthesis. TSMC 65nm library is used for the synthesis.

Introduction

The rapid growth of portable electronic devices with limited power and area has opened a vast area of low-power and compact circuit design opportunities and challenges for VLSI circuit designers. Cellular phones, PDAs, and smart cards are examples of portable electronic products that are becoming an integral part of everyday life. The popularity of these devices necessitates special considerations for their security subsystems. Unlike computer network security systems that impose less stringent limitations on the area and power consumption but put more emphasis on high throughput (several Gigabit/s), portable applications demand security hardware with more restrictions on area and power and less on throughput (several hundred kilobit/s to a few Megabit/s). This difference in requirements dictates a different approach in the design and implementation of the security systems for these devices.

Since next-generation, portable electronic devices will be used for a wide range of applications their security system must implement both private (symmetric) and public (asymmetric) key algorithms, to accommodate various application requirements.

Private Key algorithms with high throughput are suitable for data communication, while public key algorithms with much lower throughput are suitable for private key exchange and authentication. Among all available algorithms, data encryption standard (DES), advanced encryption standard (AES), and elliptic curve cryptography (ECC), which are approved by standards organizations [1]–[3], are selected for this application. DES, for past compatibility, and AES, for high security and throughput, are the major candidates for private key algorithms, and ECC is the best candidate for the public key algorithm for its encryption efficiency.

A cryptography system can be implemented in either software or hardware. Software implementations allow multiple algorithms to be supported on the same hardware platform, but they are usually slow and cannot meet the required specifications. Moreover, they are considered to be more vulnerable to side-channel attacks compared to other implementations. Side-channel attacks use physical measurements on the device, for example, the power consumption of the processor, to detect the encryption/ decryption key [4]–[6].

This paper introduces, a synthesis implementation of a cryptography coprocessor (crypto-processor) that implements two standard private-key algorithms (DES and AES), and one standard public-key algorithm (ECC), in a single design

Problem Statement

In this section, a brief introduction of the three implemented Algorithms is presented.

Data Encryption Standard (DES)

This is a well-established algorithm that has been used for more than two decades (since 1977) in military and commercial data exchange and storage. The algorithm is designed to encipher and decipher blocks of data consisting of 64 b using a 56-b key. A block to be enciphered is subjected to an initial permutation (IP), then to 16 rounds of a complex key-dependent permutation, and, finally, to another permutation which is the inverse of the IP , as shown in Fig. 1. The function f() in this figure is the heart of this algorithm and consists of an expansion, XOR, lookup table (LUT), and permutation, as depicted in Fig. 2. To decipher, it is necessary to apply the very same algorithm to an enciphered message block, using the same key.

Since the processing power of current computers is much higher than those of two decades ago, a brute-force attack (checking all possible key combinations to decipher an encrypted ciphertext) to this algorithm is possible in a relatively short time (possibly as short as a few minutes). For this reason, this algorithm is no longer considered to be a secure algorithm for many applications by the National Institute of Standards and Technology (NIST). A more secure algorithm based on DES which is still supported by NIST is called the triple data encryption algorithm (Triple DES, 3DES, or TDEA) depicted in Fig. 3. In this figure, DES represents encryption and DES^{-1} represents decryption. 3DES involves applying DES, then, DES^{-1} followed by a final DES to the plain text using three different

Tele:
E-mail addresses: aravindkumarmtech@gmail.com

key options [1], which results in a cipher text that is much harder to break.

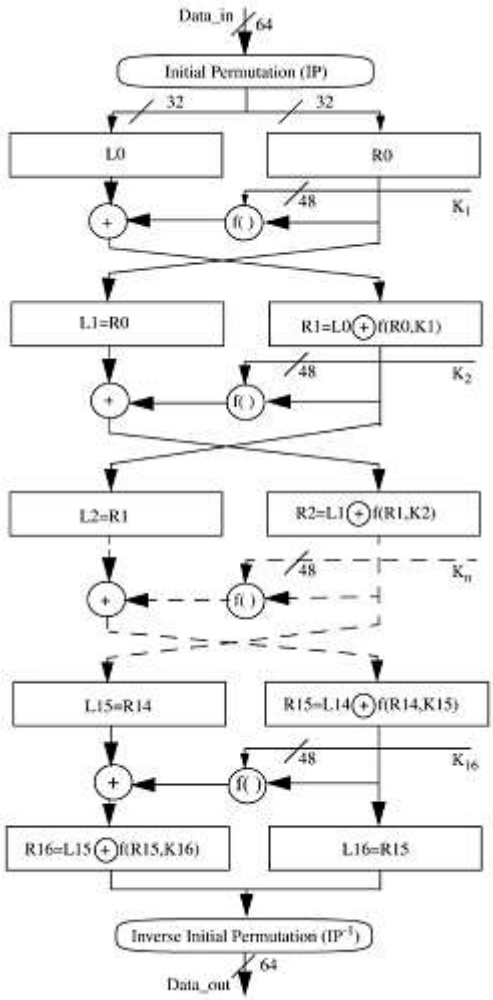

Fig 1. DES block diagram (K1, K2,. . ., K16 refer to the key values used in rounds 1–16)

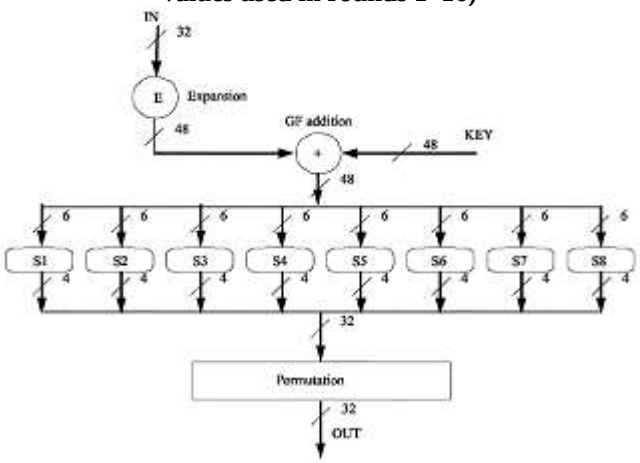

Fig. 2. DES f()-box details

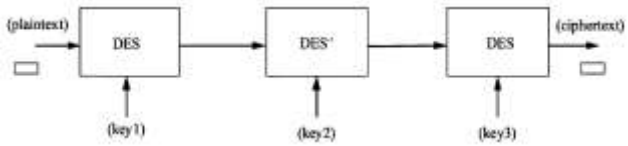

Fig 3. DES (TDEA) block diagram.

The implementation of DES needs four basic operations only, namely, the XOR, shift, LUT, and permutation, which are relatively simple to implement in hardware. The TDEA also uses the same set of operations as DES.

Advanced Encryption Standard (AES)

AES, also known as Rijndael, is a block encryption algorithm which encrypts blocks of 128 b using a unique key for both encryption and decryption [2]. A block diagram representation of the algorithm is shown in Fig. 4.

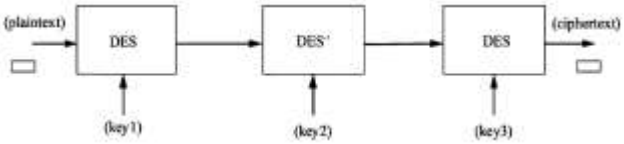

Fig. 1 3DES (TDEA) block diagram.

The implementation of DES needs four basic operations only, namely, the XOR, shift, LUT, and permutation, which are relatively simple to implement in hardware. The TDEA also uses the same set of operations as DES.

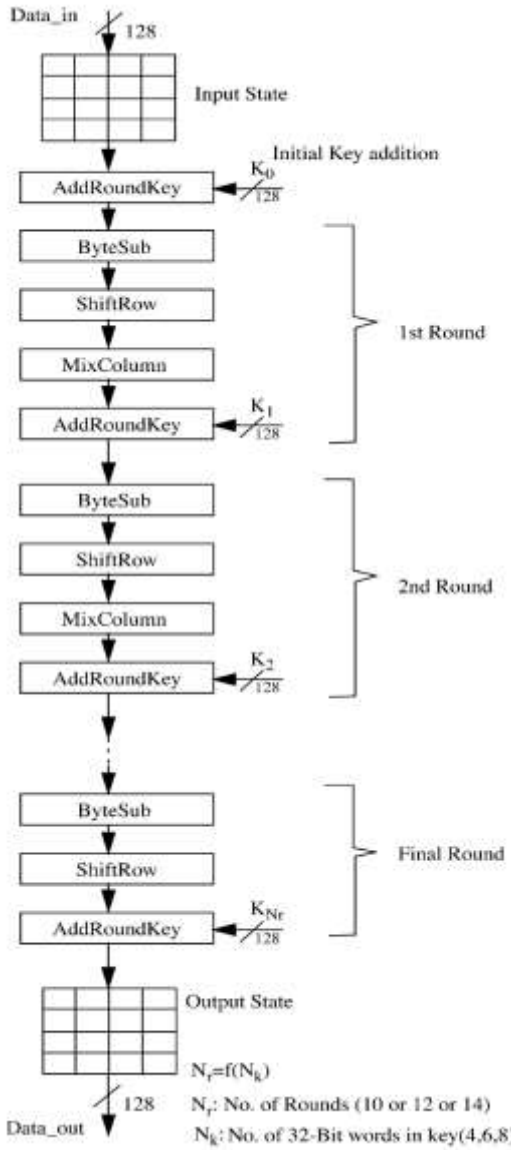

Fig 4. AES block diagram.

Three versions of the algorithm are available differing only in the key generation procedure and in the number of rounds the data is processed for a complete encryption (decryption) [2]. AES-128 uses a 128-b key and needs 10 rounds. AES-192 and AES-256, respectively, need 192-b and 256-b keys and 12 and 14 rounds for processing a block of data.

The 128-b input data is considered as a 4x4 array of 8-b bytes (also called “state” in the algorithm). The state undergoes

four different operations in each round, except for the final round which has only three operations. These operations are "*ByteSub*," "*ShiftRow*," "*MixColumn*," and "*AddRoundKey*" operations. "*MixColumn*" is omitted in the final round. Each round of the algorithm needs a 128-b key, which is generated from the input key to the algorithm. The key-scheduler block (not shown in Fig. 4) consists of two sections: the key expansion unit, which expands the input key bits to the maximum number of bits required by the algorithm, and the key selection unit, which selects the required number of bits from the expanded key, for every round [2]. As mentioned before, aside from the key values, all of the steps in all of the rounds are the same except for the last round that *MixColumn* is not present.

Each byte in the state matrix is an element of a Galois Field GF(2^8),and all of the operations can be expressed in terms of the field operations [2]. In simple terms, GF(2^n) is a set of 2^n elements each represented by an -bit string of 0's and 1's and two basic operations: addition and multiplication. These two operations are defined such that the closure, associativity, and other field properties are satisfied [7].

From the implementation point of view, *ByteSub* operation can be implemented by LUT. The *ShiftRow* can be implemented using a circular shifter. The *MixColumn* is the most complicated operation in this algorithm and needs GF(2^8) field multiply and add operations. Due to the specific choices of the parameters of the algorithm, this operation can be expressed as a matrix multiplication, which can be implemented using shift and XOR operations. A more detailed analysis of the implementation options of this block are presented in [8]. *AddRoundKey* is just a logical XOR operation.

Elliptic Curve Cryptography (ECC)

The set of all (x,y) pairs satisfying the nonsupersingular elliptic curve equation

$$y^2+xy=x^3+a_2x^2+a_6$$

are called points on the elliptic curve **E**, where x,y,a_2 and a_6 are elements of the GF(2^n). The point addition (S=P+Q) and multiplication (R=*k*.P, where *k* is a constant) operations are defined such that both S and R are also points on the elliptic curve E. Moreover, knowing R and P, it is pratically impossible to find *k*. This property forms the fundamental foundation of ECC [9].

Elliptic curves can be used in different forms in cryptography. As an example, we will explain one of the basic applications, which is the secret key exchange. The basic secret-sharing algorithm, also known as the Diffie–Hellman protocol for key exchange, is pictured in Fig. 5. [10] [11][12] In brief, both users,A and B, agree on the elliptic curve E, a point P on E, and a mathematical basis, such as polynomial basis or normal basis (NB). Each user then chooses a secret key from GF(2^m), K_a and K_b and calculates her/his own public key ($PK_a=K_a$.P and $PK_b=K_b$.P) and sends it to the other user. At this point both users can calculate the secret point S(X_s,Y_s)

$S(X_s,Y_s) = K_a.PK_b = K_b.PK_a = K_a.K_b.P$

Note that, although both X_s and Y_s are available, only one of them should be used for higher security [13]

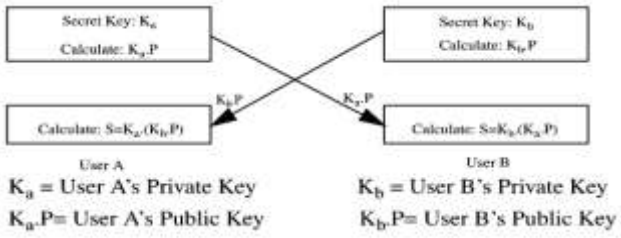

Fig 5. ECC secret-key-exchange algorithm block diagram

Cryptoprocessor Architecture

We are proposing the cryptoprocessor depicted in Fig. 6,. Depending up on the control signal and the key value the AES or DES or ECC will be selected and operated. Completed AES, DES and ECC algorithms are designed in the cryptoprocessor

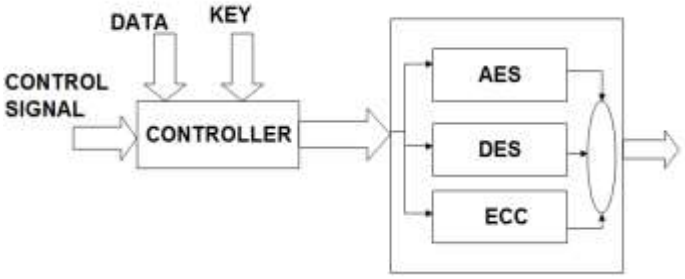

Fig 6. Detailed cryptoprocessor architecture

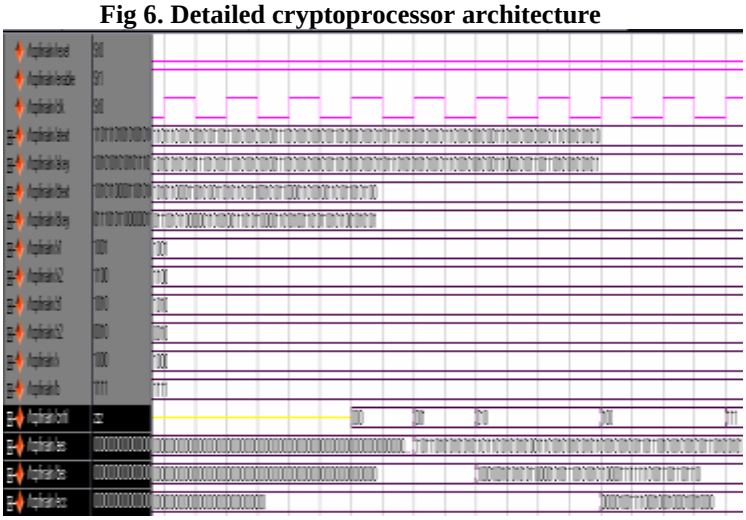

Fig. 7. Simulated waveform results of crytoprocessor (AES,DES,ECC algorithms)

Implementation Results

This section highlights the simulation and synthesis of cryptoprocessor architecture used for implementation of DES, AES, and ECC algorithms.

The algorithms were designed in verilog HDL. The simulated waveform of the cryptoprocessor is shown in fig

The cryptography processor is targeted to 65nm TSMC library. The result obtained after synthesis are shown in form of table.

Table 1. Results obtained for synthesis using TSMC 65nm library

Total Dynamic Power	Cell Leakage Power	Total cell area	Timing slack (MET)
476.3306 uW	1.0324 mW	47353.32065	0.18ns

Conclusion

In this paper we have successfully designed the cryptoprocessor. The simulation and synthesis of cryptoprocessor are presented. We have tabulated the power area and timing results obtained after synthesis.

Acknowledgment

We would like to acknowledge the "Center for VLSI Design", BVRIT to enable me to purse our research work.

References

[1] Data Encryption Standard (DES), Oct. 1999. Fed. Inf. Process. Stan- dards Pub..

[2] Advanced Encryption Standard (AES), Nov. 2001. Fed. Inf. Process. Standards Pub.

[3] IEEE Standard Specifications for Public-Key Cryptography, Jan. 2000.

[4] T. S.Messerges, E. A. Dabbish, and R. H. Sloan, "Investigation of power analysis attacks on smartcards," in Proc. USENIXWorkshop Smartcards Technology, Chicago, IL, May 1999, p. 151 and 161.
[5] K. Okeya and K. Sakurai, "A multiple power analysis breaks the ad- vanced version of the randomized addition-subtraction chains counter- measure against side channel attacks," in Proc. IEEE Inf. Theory Work- shop, 2003, pp. 175–178.
[6] S. B. Ors, F. Gurkaynak, E. Oswald, and B. Preneel, "Power-analysis attack on an ASIC AES implementation," in Proc. Inf. Technol.: Coding Computing, vol. 2, 2004, pp. 546–552.
[7] N. Biggs, Discrete Mathematics. Oxford, U.K.: Oxford Univ. Press, 2002.
[8] X. Zhang and K. K. Parhi, "Implementation approaches for the advanced encryption standard algorithm," IEEE Circuits Syst. Mag., vol. 2, no. 4, pp. 24–46, Apr. 2002.
[9] P. H. W. Leong and I. K. H. Leung, "A microcoded elliptic curve pro- cessor using FPGA technology," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 10, no. 5, pp. 550–559, Oct. 2002.
[10] J. H. Kim and D. H. Lee, "A compact finite field processor over GF(2 m) for elliptic curve cryptography," in Proc. ISCAS, vol. 2, pp. 340–343.
[11] M. Rosing, Implementing Elliptic Curve Cryptography. Greenwich, CT: Manning, 1999
[12] I. Blake et al., Elliptic Curves in Cryptography. Cambridge, U.K.: Cambridge Univ. Press, 1999.
[13] A.Menesez, Elliptic Curve Public Key Cryptosystems. Dordrecht, The Netherlands: Kluwer, 1993, ch. 6, pp. 83–99.