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.

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 65 nm library is used for the synthesis.

© 2014 Elixir All rights reserved
key options [1], which results in a cipher text that is much harder to break.

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.

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.

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^n), 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 n-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^n) 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

\[ y^2 + xy = x^3 + ax + a_6 \]

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 = kP, 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 practically 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^n), \( K_a \) and \( K_b \) and calculates her/his own public key \( PK_a = K_aP \) and \( PK_b = K_bP \) and sends it to the other user. At this point both users can calculate the secret point \( S(X, Y) = K_aPK_b = K_bPK_a = K_aK_bP \).

Note that, although both \( X \) and \( Y \) are available, only one of them should be used for higher security [13].

![Fig 5. ECC secret-key-exchange algorithm block diagram](image)

**Cryptoprocessor Architecture**

We are proposing the cryptoprocessor depicted in Fig. 6. Depending 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](image)

**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**

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Total Dynamic Power (μW)</th>
<th>Cell Leakage Power (mW)</th>
<th>Total cell area (MET)</th>
<th>Timing slack (MET)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AES</td>
<td>476.3306</td>
<td>1.0324</td>
<td>47353.32065</td>
<td>0.18ns</td>
</tr>
<tr>
<td>DES</td>
<td>65nm TSMC</td>
<td>1.0324</td>
<td>47353.32065</td>
<td>0.18ns</td>
</tr>
</tbody>
</table>

**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”, BVRRIT to enable me to pursue our research work.

**References**


