Optimized Area and Optimized Speed Hardware Implementations of AES on FPGA

M.R.M. Rizk, Senior Member, IEEE and M. Morsy

ABSTRACT

The Advanced Encryption Standard (AES) is the last standard for cryptography and has gained wide support as means to secure digital data. In this paper, Tradeoffs of speed vs. area that are inherent in the design of a security processor are explored. Two implementations of the AES on Xilinx Virtex 4 FPGA are introduced, the first design is called optimized area AES which is based on the basic architecture of the AES, the second one is called optimized speed AES which is based on the sub-pipelined architecture of the AES. An AES crypto processor with serial interface was implemented and it could be used with any of our designed encryptor or decryptor.

Keywords: encryption, security processor, architecture, FPGA.

INTRODUCTION

On November 26, 2001, the algorithm known as Rijndael was chosen to be the replacement for DES and since then it is known as the Advanced Encryption Standard (AES). The AES has been the topic of much research to find suitable architectures for its hardware implementation [1]. Architectural choices for a given application are driven by the system requirements in terms of speed and the resources consumed. This can simply be viewed as throughput and area. In this brief, we introduce two optimized hardware implementations of the AES encryptor and decryptor with 128-bit plaintext and 128-bit key on Virtex 4 FPGA and a crypto processor with serial interface which could be used with our implementation of AES Algorithm. The first hardware implementation of the AES is targeted to the applications which required a high security with minimum resources such as smart cards and we call this implementation optimized area AES. The second hardware implementation of the AES is targeted to the applications which required a high output throughput such as network routers and we will call this implementation optimized speed AES. The optimized area AES design is based on the basic architecture of the AES [2] while the optimized speed AES design is based on the pipelined architecture [2]. For the two designs we make an algorithmic optimization for the internal transformations depend on the goal of the design (speed or area).

AES ALGORITHM

An AES encryption process for a 128-bit plain text data block and a 128-bit secret key is shown in Fig 1. A sequence of four primitive functions: SubBytes, ShiftRows, MixColumns and AddRoundKey are executed N_r times. Each loop is called a round and the concrete value of N_r is 10, 12 or 14 depending on the key length. Prior to this main loop, AddRoundKey is executed for initialization. After executing the main loop, a sequence of SubBytes, ShiftRows and AddRoundKey is executed as the final round [3, 4].

SubBytes is a 16-B (128-bit) input/output nonlinear transformation that uses 1-B Substitution table (S-Boxes). Each S-Box is a multiplicative inversion over Galois field GF (2^8) followed by an affine transformation. The irreducible polynomial used by the field is

$$m(x) = x^8 + x^4 + x^3 + x + 1 \tag{1}$$

The affine transformation is defined by the following equation

$$\begin{bmatrix} b_0' \\ b_1' \\ b_2' \\ b_3' \\ b_4' \\ b_5' \\ b_6' \\ b_7' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \tag{2}$$

ShiftRows is a cyclic shift operation in each row of a 4×4-byte data block by 0-3 B offsets. MixColumns treats the 4-B data in each column as a four-term polynomial, and multiplies the data modulo $x^4 + 1$ with a fixed polynomial given by

$$a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\} \tag{3}$$

AddRoundKey is a simple bitewise XOR operation on the 128-bit round sub-keys and the data.

In the decryption process, the inverse operations of each primitive function are executed. The inverse of AddRoundKey is AddRoundKey itself.

InvSubBytes, which is the inverse of SubBytes, executes an affine transformation defined by (4) before the multiplicative inversion.

$$\begin{bmatrix} b_0' \\ b_1' \\ b_2' \\ b_3' \\ b_4' \\ b_5' \\ b_6' \\ b_7' \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \tag{4}$$

InvShiftRows is a cyclic rotation in the reverse direction. InvMixColumns uses the following polynomial for the multiplications:

$$a^{-1}(x) = \{0b\}x^3 + \{0d\}x^2 + \{09\}x + \{0e\} \tag{5}$$

The complete AES encryption and decryption process are shown in Fig 1.

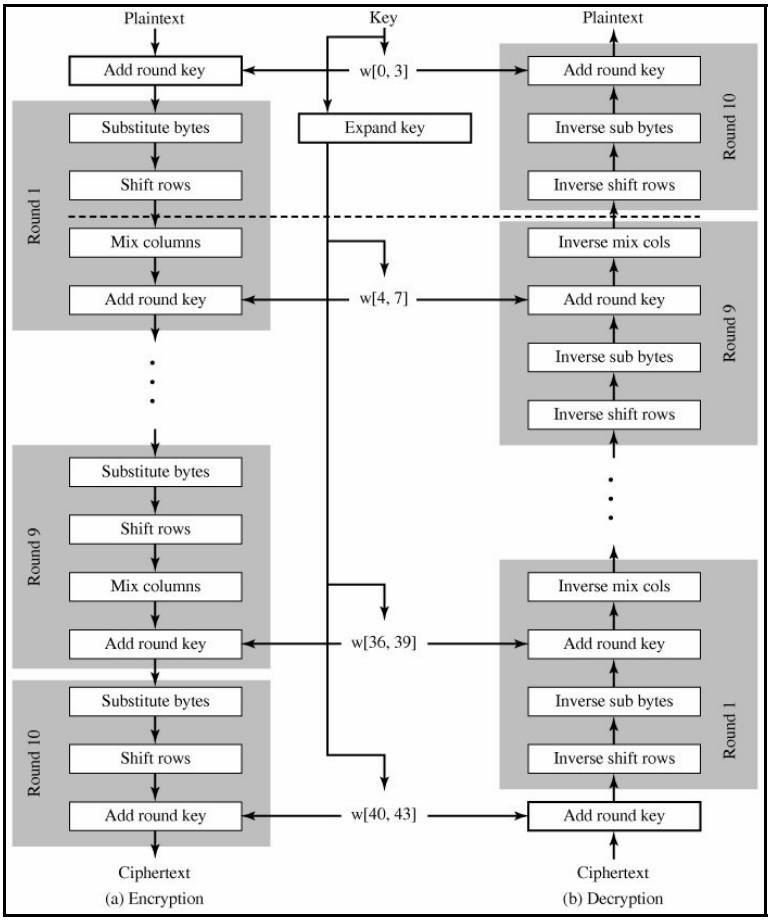


Figure 1: AES Encryption and Decryption

The 128-bit AES key expansion algorithm takes as input a 4-word (16-byte) key and produces a linear array of 44 words (176 bytes) as shown in Fig 2. This is sufficient to provide a 4-word round key for the initial AddRoundKey stage and each of the 10 rounds of the cipher.

The function g consists of the following sub-functions:

1. RotWord performs a one-byte circular left shift on a word. This means that an input word $[b_0, b_1, b_2, b_3]$ is transformed into $[b_1, b_2, b_3, b_0]$.
2. SubWord performs a byte substitution on each byte of its input word, using the S-box (Table 2–2-a).
3. The result of steps 1 and 2 is XORed with a round constant, *Rcon[j]*.

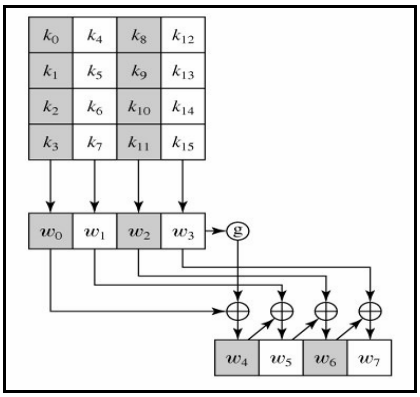


Figure 2: AES key expansion

Optimized Area AES Encryptor/ Decryptor

In our Implementation of the AES algorithm with a small area we will select the basic reference architecture shown in Fig 3 which needs the implementation of one round only and re-use it to complete the ten encryption rounds, we also designed the Encryptor/ Decryptor to complete one encryption round in one clock cycle so the output of the Encryptor/ Decryptor will be valid after ten clock cycles from the data entrance.

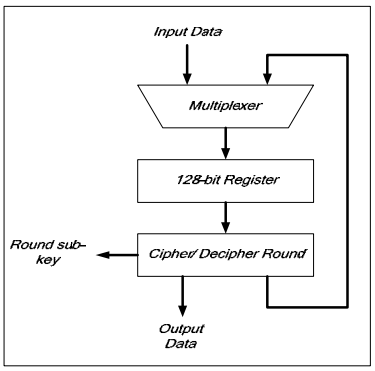


Figure 3: Optimized area cipher /decipher Architecture

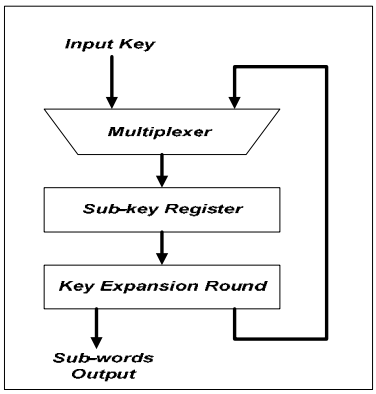


Figure 4: Cipher key expansion architecture

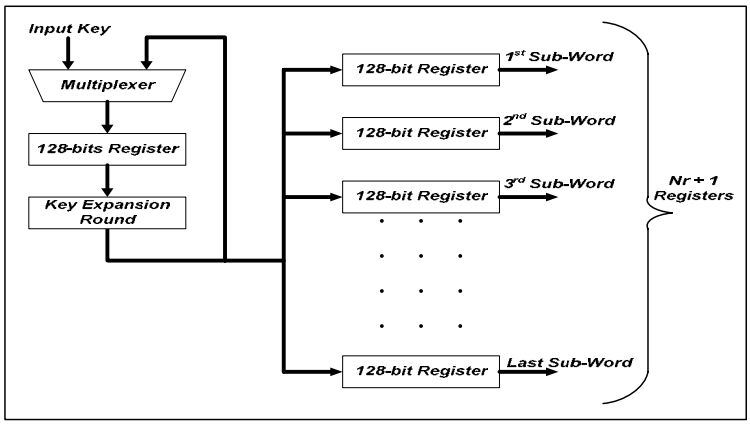


Figure 5: Decipher key expansion architecture

The key schedule architecture is chosen to generate all the sub-keys on the fly in parallel with the encryption module. For the encryptor we implement the hardware required to generate one set of sub-key and re-use it in the calculation of the other sub-keys, and at the same time also use one clock cycle for one sub-key generation as shown in Fig 4. For the decryptor we must generate the last sub-key first to use it in the first decipher round, so we couldn't use the same key expansion architecture used with cipher and we must select one of the other architectures either by generation of all sub-keys beforehand and storing them in a RAM as shown in Fig 5.

No optimization can be performed on ShiftRows/ InvShiftRows and AddRoundKey transformations, since no logic gates are needed for the former transformation and only one step of XOR operation is needed for the latter. However, different methods can be used to implement the SubBytes/ InvSubBytes and MixColumns/InvMixColumns transformations. For the SubBytes transformation we use the composite field method described in [5, 6] which will save the area but it will increase the delay. For the MixColumns transformation we use the method of

substructure sharing described in [7, 8] in our implementation for this transformation which has the advantages of low area and high speed.

Fig 6 shows the encryptor top level entity and Fig 7 and 8 show encryptor and decryptor simulation results.

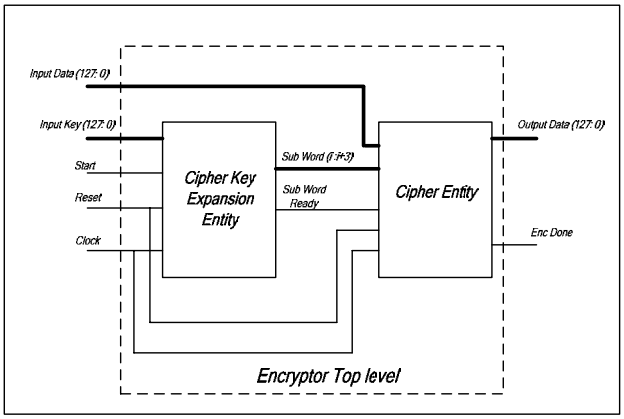


Figure 6: Encryptor top level entity

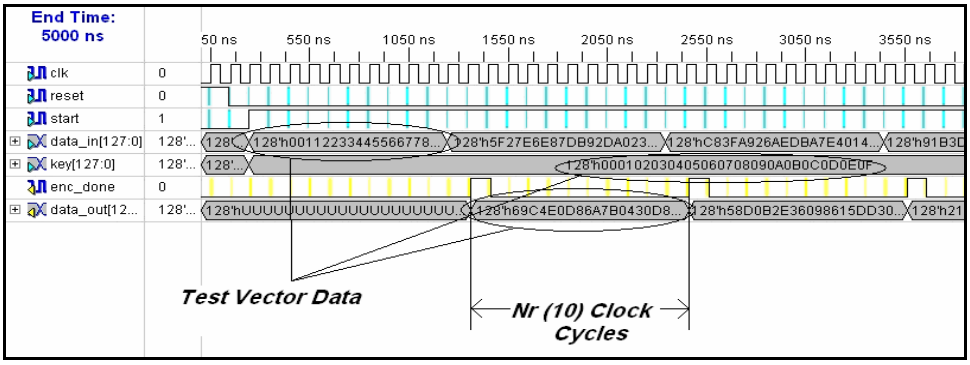


Figure 7: Optimized area AES Encryptor simulation results

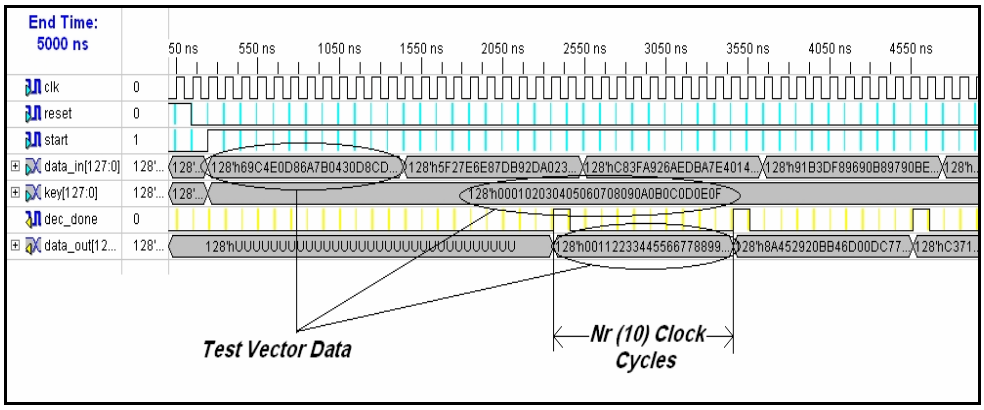


Figure 8: Optimized area AES Decryptor simulation results

Optimized Speed AES Encryptor/ Decryptor

In our Implementation of the AES algorithm with a high speed we will select the pipelined architecture with (*K=Nr=10*) ten rounds shown in Fig 9. This design will allow to us to update the input data each clock cycle but it will increase the area about ten times larger than optimized area AES. The key schedule architecture for the encryptor and decryptor is chosen to generate all the sub-keys beforehand and storing

them in a buffer, or by generation of all sub-keys using pipelined architecture as shown in Fig 10.

We will implement all transformations in the same methods like optimized area AES except the SubBytes/ InvSubBytes Transformation which will be implemented using the Look Up Table (ROM) method [9] to decrease the delay.

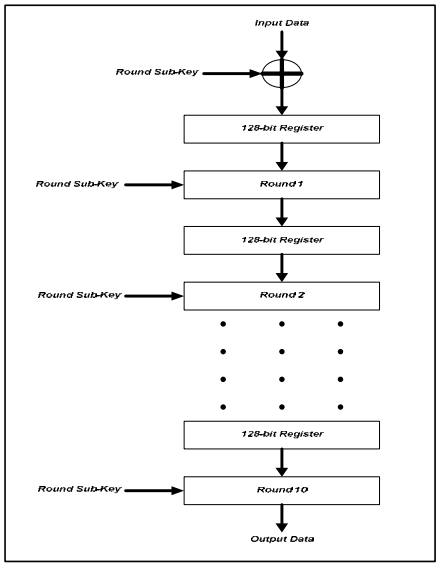


Figure 9: Optimized Speed Cipher/ Decipher Architecture

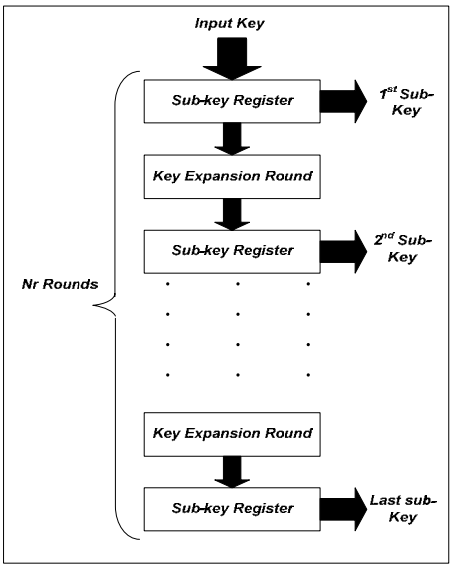


Figure 10: Optimized speed Key Expansion Architecture

Fig 11 shows the encryptor top level entity and Fig 12 and 13 show encryptor and decryptor simulation results.

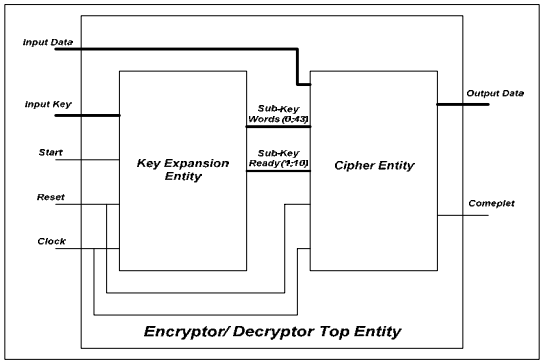


Figure 11: Encryptor/ Decryptor top level entity

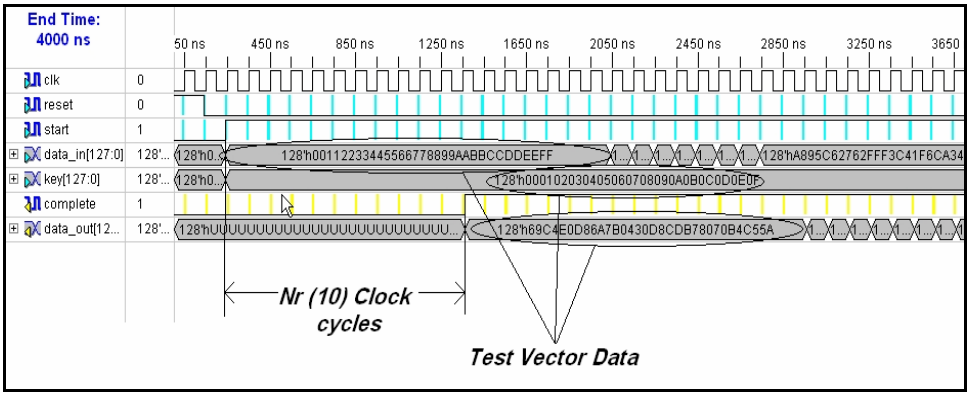


Figure 12: Optimized speed AES Encryptor simulation results

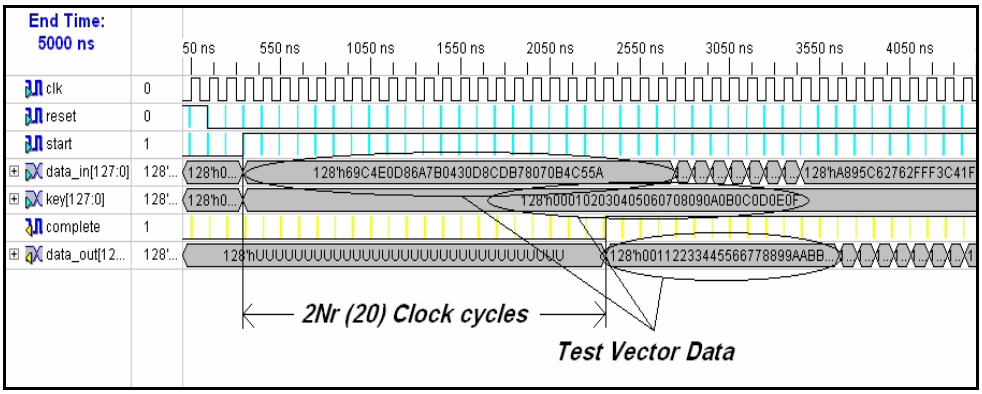


Figure 13: Optimized speed AES Decryptor simulation results

Comparison of some Related Work for FPGAs

There are many modes of operation of the AES block cipher, and these modes are classified into two major classes: feedback and non-feedback modes [], in our design we will concern in non feedback mode of operation ECB (Electronic code book)

Our hardware designs have been encoded in VHDL'93, and targeted on a Xilinx Virtex 4 (4vlx60ff668-12) FPGA. We use Xilinx ISE 7.1i and modelsim programs for simulation, synthesis, place and route for my designs.

The architecture of an AES implementation mainly defines the required hardware resources on an FPGA. Additionally, the used synthesis tool and the target device influence this result.

Table 1 gives an overview of existing FPGA solutions. Because of the different FPGAs, most of the use Xilinx FPGAs, the values have to be seen as a relative comparison of resource requirements and data throughput [10].

Table 1: Comparison between difference FPGA implementations of AES

Authors	*LUTs*	*Block RAMs*	*Throughput [Gbps]*
Chodowiec	*222*	*3*	*0.166*
Chodowiec	*12,600*	*80*	*12.16*
Chodowiec	*2,057*	*8*	*1.265*
Chodowiec	*2,507*	*0*	*0.414*
Hodjat	*9,446*	*0*	*21.64*
Hodjat	*5,177*	*84*	*21.54*
McLoone	*2,222*	*100*	*7.0*
Pramstaller	*1,125*	*0*	*0.215*
Rouvroy	*146*	*3*	*0.358*
Saggese	*446*	*10*	*1.0*
Saggese	*648*	*10*	*1.82*
Saggese	*2,778*	*100*	*8.9*
Saggese	*5,810*	*100*	*20.3*
Standaert	*1,769*	*0*	*2.085*
Standaert	*15,112*	*0*	*18.560*
Wang	*1,857*	*0*	*1.604*
Zambreno	*387*	*10*	*1.41*
Zambreno	*1,254*	*20*	*4.44*
Zambreno	*2,206*	*50*	*10.88*
Zambreno	*3,766*	*100*	*22.93*
Zambreno]	*16,938*	*0*	*23.57*
Zhang	*9,406*	*0*	*11.965*
Zhang]	*11,022*	*0*	*21.556*
Our optimized area encryptor	*1468*	*0*	1.664
Our optimized area decryptor	*2752*	*0*	1.598
Our optimized speed encryptor	*18855*	*200*	28.51
Our optimized speed Decryptor	*20155*	*200*	23.09

AES Crypto Processor

In this section we will introduce a simple processor that could be used to make the interface between the implemented AES encryptor/ decryptor datapaths and other external peripheral under the control of an operator. We introduce two modes of operation in the AES crypto processor. We called the first mode of operation discrete mode in which all the data operations (input, output and processing) could be done by orders from the operator. The second mode of operation is called the continuous mode in which the operator will only start to get the key and all the consequent operation will be done sequentially. We will use the first mode of operation to make the timing simulations (post synthesis, post map and post place and route simulations) and practical tests to the implemented hardware.

Crypto Processor Hardware Circuit

The crypto processor is mainly consists of the following components (Fig14):

1. Encryptor/ decryptor unit: Any one of the previous implemented encryptors and decryptors could be used in the crypto processor.
2. Input interface unit: It is a serial interface with handshaking between the processor and external peripheral which is used to get the 128-bit data input (plaintext/ ciphertext) to the encryptor/ decryptor and it is mainly consists of serial to parallel shift register which takes data each clock cycle (it will takes 128 clock cycles to complete the data input to encryptor/ decryptor) and it has the start, complete and reset as asynchronous control signals.
3. Key interface unit: This component is similar to the input interface unit and it is used to input the 128-bits key used in encryption/ decryption unit.
4. Output interface unit: It is a 128-bit parallel to serial converter which is used to output the data (ciphertext/ plaintext) serially from the encryptor/ decryptor unit. Similar to the input and key interface units, the output interface unit takes 128 clock cycles to output the data and it has the start, complete and reset as asynchronous control signals.
5. Control Unit: It is a Moore finite state machine FSM (see Figure 6–19) which forms the interface between the operator and all another units in the processor. The control unit is used to generate all asynchronous control signals for all units in the design. From the Figure 6–19, it is clear that we will use the same states in the FSM for both of the two modes (continuous mode with the dotted arrows and discrete mode with solid arrows).

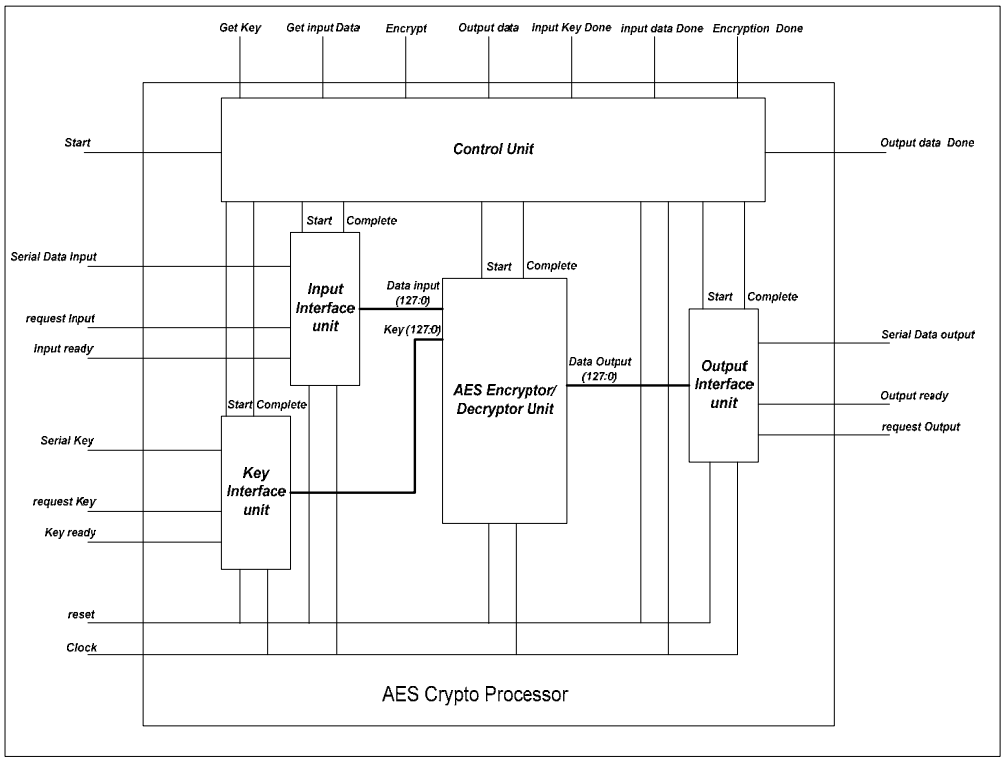
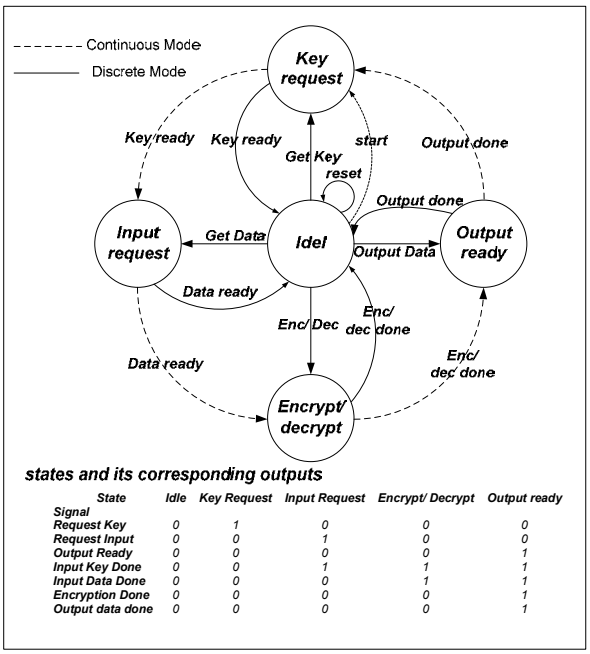


Figure 14: AES crypto processor



State / Signal	Idle	Key Request	Input Request	Encrypt/ Decrypt	Output ready
Request Key	0	1	0	0	0
Request Input	0	0	1	0	0
Output Ready	0	0	0	0	1
Input Key Done	0	0	1	1	1
Input Data Done	0	0	0	1	1
Encryption Done	0	0	0	0	1
Output data done	0	0	0	0	1

Figure 15: Control Unit FSM

Crypto Processor Functional Simulation Results

Fig 16 shows the functional simulation for the AES crypto processor with the optimized area AES encryptor in the discrete mode of operation.

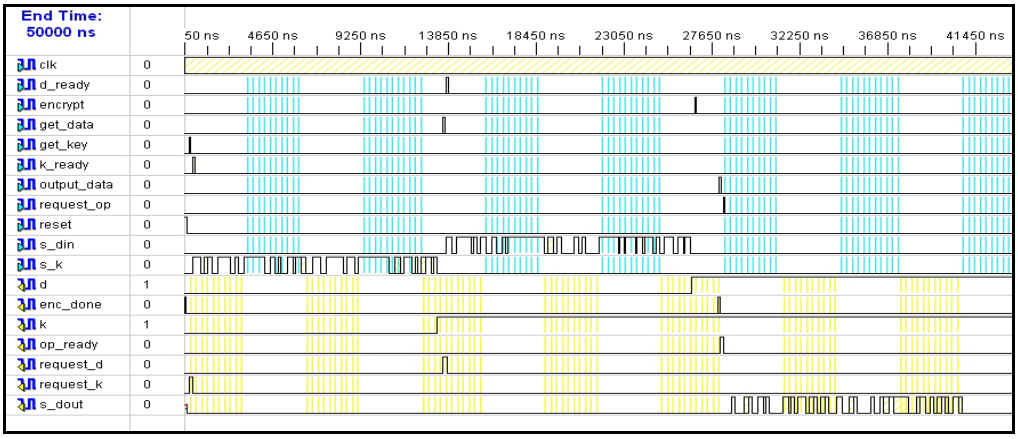


Figure 16: Behavioral simulation of AES crypto processor

Crypto Processor Timing Simulation Results

Fig 17 shows the post place and route simulation results which agrees with the functional simulation results and the following message appears on the modelsim simulator screen:

```
** Warning: /X_LATCHE RECOVERY  Low VIOLATION ON SET WITH RESPECT TO CLK;
#   Expected := 0.606 ns; Observed := 0.072 ns; At : 1.686 ns
#    Time: 1686 ps  Iteration: 5  Instance: /proc_test/uut/c7_dout_ok_4027
# ** Failure: Simulation successful (not a failure).  No problems detected.
#    Time: 50100 ns  Iteration: 0  Process: /proc_test/line__115 File: proc_test.timesim_vhw
```

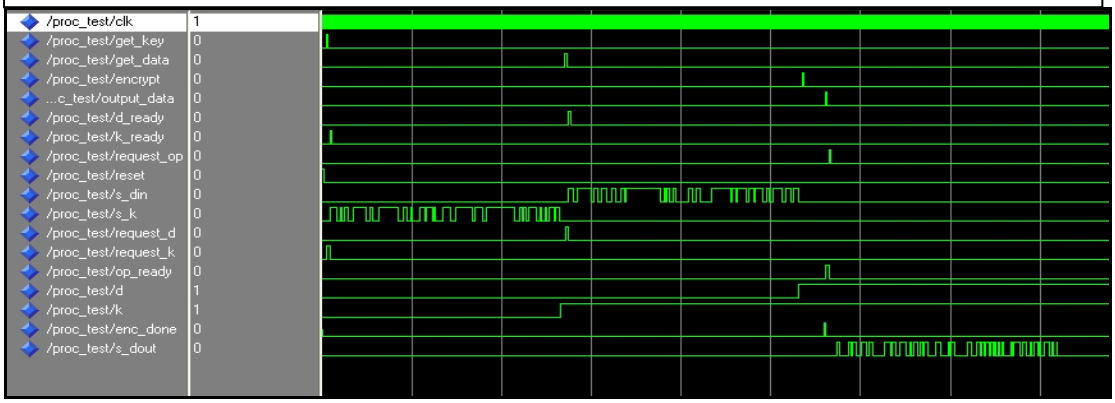


Figure 17: post place and route simulation of AES crypto processor

Similarly we have made the timing simulation for the optimized area decryptor and optimized speed encryptor and decryptor and we got results as same as the above simulation.

Conclusion

Cryptography plays an important role in the security of data transmission. Different applications of the AES algorithm may require different speed/area trade-offs. Our work aims to implement a low area and a high speed AES encryptor and decryptor using various optimization techniques and to implement AES crypto processor with serial interface with external peripherals on FPGA. These goals have been met. Optimized area and optimized speed AES encryptor and decryptor and AES crypto processor are completed, simulated and verified. The code was written in

VHDL'93 and synthesized and verified using the Xilinx ISE 7.1 program and simulated using the Modelsim program.

Optimized area AES (encryptor, decryptor) have been implemented based on the basic architecture and it consumes *(1468, 2752 Xilinx slices)* and operates at *(1.664, 1.558 Gbps)*. Optimized speed AES (encryptor, decryptor) have been implemented based on the basic architecture and it consumes *(18855, 20155 Xilinx slices)* and operates at *(28.51, 23.09 Gbps*), which was greater than other works cited in this article.

References

[1] S. William, "Cryptography and Network Security Principles and Practices", Fourth Edition, Prentice Hall, November 16, 2005.

[2] Z. Xinamiao and K. Parhi, "Implementation Approaches for The Advanced Encryption Standard", Circuit and System Magazine, Volume 2, Number 4, Fourth Quarter 2002.

[3] "Advanced Encryption Standard(AES)", Federal Information Processing Standards Publication 197, November 26, 2001

[4] J. Daemen and R.Rijmen, "AES Proposal: Rijndael", version 2, 1999. http://www.esat.kuleuven.ac.be/~rijmen/ rijndael.

[5] A.Rudra, P. K. Dubey, C. S. Jutla, V. Kumar, J. R. Rao, and P. Rohatgi, "Efficient Implementation of Rijndael Encryption with Composite Field Arithmetic", Proceedings CHES 2001, pp. 171–184, Paris, France, May 2001.

[6] V. Rijmen, "Efficient Implementation of the Rijndael S-box", http:// www.esat.kuleuven.ac.be/~rijmen/rijndael.

[7] C. C. Lu and S. Y. Tseng, "Integrated Design of AES (Advanced Encryption Standard) Encrypter and Decrypter", IEEE Transactions on Information Theory, vol. 37, no. 5, pp. 1241–1260, September 1991.

[8] V. Fischer, "Realization of the Round 2 Candidates Using Altera FPGA", The Third AES Conference (AES3), New York, Apr. 2000. http://csrc.nist.gov/encryption/aes/round2/conf3/aes3papers.html.

[9] M. McLoone and J. V. McCanny, "Rijndael FPGA Implementation Utilizing Look-Up Tables", IEEE Workshop on Signal Processing Systems, pp. 349–360, September 2001.

[10] E. Oswald, "State of the Art in the Hardware Architecture", European Network of Excellence in Cryptology, 2005.

Contact 1st Author: Prof. Dr. Mohamed R. M. Rizk

EE Dept., Faculty of Eng., Alexandria University

Hadara, Alexandria, Egypt.

Phone Number: +20 101545412

e-mail: mrmrizk@ieee.org

Contact 2nd Author: Eng. Mohammed Morsy

EE Dept., Faculty of Eng., Alexandria University

Hadara, Alexandria, Egypt.

Phone Number: +20 106047930

e-mail: m-morsy@alex.edu.eg