ISSN:2320-0790

An International Journal of Advanced Computer Technology

Area Optimized Advanced Encryption Standard

Mr. Sandip R. Aher, Prof. Dr. G. U. Kharat

Dept. of Electronics and Telecommunication Engineering
SPCOE, Otur, University of Pune
Pune, Maharashtra, India

***Abstract*— Performance evaluation of the Advanced Encryption Standard candidates has become led to intensive study of both hardware and software implementations. However, number of papers presents various implementation results, it shows that efficiency could still be greatly improved by applying effective design rules adapted to devices and algorithms. This paper shows various approaches for efficient FPGA implementations of the Advanced Encryption Standard algorithm. For different applications of the AES algorithm may require different speed/area tradeoffs, we propose a vital study of the possible implementation schemes, but also the discussion of design methodology and algorithmic optimization in order to improve previous reported results. We propose system to evaluate hardware efficiency at different steps of the design process. We also use an optimal pipeline that takes the place and route constraints into account. Resulting circuits significantly improve the previous reported results: throughput has been up to 18.5 Gbits/sec and the area requirements can be limited to 542 slices and 10 RAM blocks with a ratio throughput/area improved by minimum 25% of the best-known designs in the Xilinx Virtex- E technology.**

Keywords— *EDK, Real time* Communication, AES, Security, XPS, EDK, RTOS.

BACKGROUND

The Advanced Encryption Standard (AES) specifies a cryptographic algorithm that can be used to protect electronic data. AES algorithm is asymmetric block cipher that can encrypt (encipher) and decrypt (decipher) information. Encryption converts data to an unintelligible form called cipher-text; decrypting the cipher-text converts the data back into its original form, called plaintext.

The Advanced Encryption Standard, after the Data Encryption Standard was found too weak because of its small key size and the technological advancements in processor power. Fifteen candidates were accepted and based on public comments the pool was reduced to five. One of these five algorithms was selected as the forthcoming standard: a slightly modified version of the Rijndael.

The Rijndael, whose name is based on the names of its two Belgian inventors, Joan Daemen and Vincent Rijmen is a Block cipher, which means that it works on fixed length group of bits, which are called *blocks*. It takes an input block of a certain size, usually 128 bits, and produces a corresponding output block of the same size. The transformation requires a second input, which is the secret key with lengths of 128, 192 and 256 bits. Unlike DES, which is based on Feistel network, AES is a substitution-permutation network, which is a series of mathematical operations that use substitutions (also called S-Box) and permutations (P-Boxes) and their careful definition implies that each output bit depends on every input bit.

BLOCK CIPHER

When a ***block cipher*** algorithm is used for encryption and decryption purposes, the message is divided into blocks of bits. These blocks are then put through substitution, transposition, and other mathematical functions.

The algorithm dictates all the possible functions available to be used on the message, and it is the key that will determine what order these functions will take place. Strong algorithms make

reengineering or trying to figure out all the functions that took place on the message, basically impossible.

It has been said that the properties of a cipher should contain confusion and diffusion. Different unknown key values cause confusion, because the attacker does not know these values, and diffusion is accomplished by putting the bits within the plaintext through many different functions so that they are dispersed throughout the algorithm. Block ciphers use diffusion and confusion in their methods.

ADVANTAGES OF AES:

- Through AES, input message of length 128 bits can be encrypted which is more than the DES and Triple DES.
- AES has the various secret key lengths such as 128 bits, 192 bits and 256 bits, whereas DES and Triple DES have fixed length of 64 bits.
- The cipher key is expanded into a larger key, which is later used for the actual operation.
- The Expanded Key shall ALWAYS be derived from the Cipher Key and never be specified directly.
- AES is very hard to attack or crack when compared to DES.
- AES will be faster when compared to the Triple DES.

APPLICATION

- This standard may be used by Federal departments and agencies when an agency determines that sensitive (unclassified) information (as defined in P. L. 100-235) requires cryptographic protection
- High speed ATM/Ethernet/Fiber-Channel switches
- Secure video teleconferencing
- Routers and Remote Access Servers

AES ALGORITHM

The AES is an iterated block cipher with a fixed block size of 128 and a variable key length. The different transformations operate on the intermediate results, called *state*. The state is a rectangular array of bytes and since the block size is 128 bits, which is 16 bytes, the rectangular array is of dimensions 4x4.The basic unit for processing in the AES algorithm is a **byte,** a sequence of eight bits treated as a single entity. The input, output and Cipher Key bit sequences which are processed as arrays of bytes that are formed by dividing these sequences into groups of eight contiguous bits to form arrays of bytes.

In the Rijndael version with variable block size, the row size is fixed to four and the number of columns varies. The number of columns is the block size divided by 32 and denoted Nb. The cipher key is similarly pictured as a rectangular array with four rows. The number of columns of the cipher key, denoted Nk, is equal to the key length divided by 32. AES uses a variable number of rounds, which are fixed: A key of size 128 has 10 rounds.

AES algorithm uses a round function that is composed of four different byte-oriented transformations:

- Byte substitution using a substitution table (S-box)
- Shifting rows of the State array by different offsets
- Mixing the data within each column of the State array

Adding a Round Key to the State

Above mentioned functions were carried out for every individual round and in the last round the third function, that is, Mixing the data within each column of the State array will not be performed. Hence the last round is carried out separately. Based on the key provided, the new set of keys will be generated in the Key Expansion block and is given to the each round as input.

ENCRYPTION

At the start of the Encryption or Cipher, the input data and the input key were copied to the State array using the conventions. Initially the XOR operation should be performed between each byte of the input data and the input key and the output will be given as the input of the Round-1. After an initial Round Key addition, the State array is transformed by implementing a round function 10times, with the final round differing slightly from the first ***Nr***–1rounds. The final State is then copied to the output. The round function is parameterized using a key schedule that consists of a one-dimensional array of four-byte words derived using the Key Expansion routine.

The individual transformations that carried out are listed below.

- SubBytes

- ShiftRows
- MixColumns
- AddRoundKey

DECRYPTION

The cipher text of 128 bits and the same key of 128 bits will be given as the input to the decryption block. The encrypted data will be decrypted and the original plain message will be achieved as the output of the decryption block. The Cipher transformations can be inverted and then implemented in reverse order to produce a straightforward Inverse Cipher for the AES algorithm. The individual transformations used in the Inverse Cipher were listed as follows.

- InvShiftRows
- InvSubBytes
- InvMixColumns
- AddRoundKey

Here also 10 rounds will be carried out and the only difference in the decryption block with respect to the algorithm flow is that the result of the KeyExpansion of each round will also be given to the MixCoulmns operation after which the AddRoundKey transformation should be carried out.

InvMixColumns (state XOR Round Key) = InvMixColumns (state) XOR InvMixColumns (Round Key)

The above equation represents the basic difference in the process of the AES Encryption and Decryption algorithm.

IMPLEMENTATION

The AES is a block cipher. This means that the number of bytes that it encrypts is fixed. AES can currently encrypt blocks of 16 bytes at a time; no other block sizes are presently a part of the AES standard. If the bytes being encrypted are larger than the specified block then AES is executed concurrently. This also means that AES has to encrypt a minimum of 16 bytes. If the plain text is smaller than 16 bytes then it must be padded. Simply said the block is a reference to the bytes that are processed by the algorithm.

The current condition of the *block* will be defined by the State. That is the block of bytes that are currently being worked on. The state starts off being equal to the block, however it changes as each round of the algorithms executes. Plainly we can say that this is the block in progress. The Advanced Encryption Standard Algorithm which includes both Encryption and Decryption are implemented using Verilog and their functionality will be verified in the ModelSim Tool with proper test cases.

IMPLEMENTATION REQUIREMENTS

During the implementation, there are different parameters are required which are discussed as follows.

- ***Input Data Length Requirements***

An implementation of the AES algorithm should have the input data (Plain Text) length of 128bits which acts as the primary input to the both Encryption and Decryption block.

- ***Key Length Requirements***

In this AES implementation the input key chosen to be as 128bits from the various key lengths available. This also acts as the primary input to the both Encryption and Decryption block.

- ***Keying Restrictions***

No weak or semi-weak keys have been identified for the AES algorithm and there is no restriction on key selection.

- ***Parameterization of Block Size and Round Number***

Here since the input data and the input key lengths are 128 bits, the block size will be of Nb = 4 and the Round Number will be of Nr = 10. The Round Number will be taken with respect to the AES Algorithm Standard.

RESULTS AND CONCLUSION

Design Summary

This describes the simulation on Xilinx navigator summary statement.

Device Utilization Summary				
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Flip Flops	551	29,504	1%	
Number of 4 input LUTs	2,116	29,504	7%	
Logic Distribution				
Number of occupied Slices	1,210	14,752	8%	

Number of Slices containing only related logic	1,210	1,210	100%	
Number of Slices containing unrelated logic	0	1,210	0%	
Total Number of 4 input LUTs	2,116	29,504	7%	
Number of bonded IOBs	133	250	53%	
Number of GCLKs	1	24	4%	
Total equivalent gate count for design	17,914			
Additional JTAG gate count for IOBs	6,384			

Performance Summary			
Final Timing Score:	0	**Pinout Data:**	Pinout Report
Routing Results:	All Signals Completely Routed	**Clock Data:**	Clock Report
Timing Constraints:	All Constraints Met		

5.2 Simulation Waveforms

1) Simulation Waveforms for Initial Round:

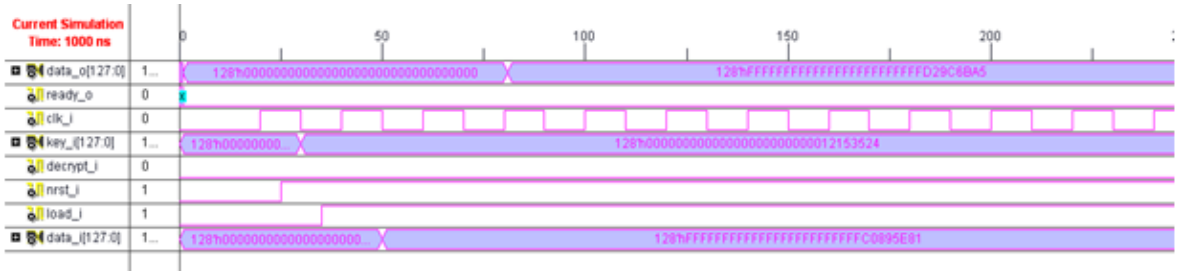
Functional Simulation and Verification

The new improved structure of AES-128 encryption algorithm is implemented with Verilog. We used Xilinx 9.2 ISE for the waveform and verified the results.

Table 1 ***Comparison of implementations of AES***

Slices	**Throughput (Gbps)**	**Throughput/Area (Mb/Sec/ Slice)**	**Support**	**Ref**
1931	-	-	Enc	[1]
22994	-	-	Enc/Dec	[4]
8447	1.18	0.187	Enc	[5]
626	3.4	5.43	Enc	[6]
1470	2.8	1.9	Enc/Dec	[7]
751	4.0	5.33	Enc	Serial Implem
951	5.25	9.16	Enc/Dec	Pipelined Implem

Fig. 1 ***Simulation Results Initial Round***



Conclusion

Efficiency in terms of architecture optimizations such as those made in the Advanced Encryption Standard and implementation aspects leading to

- ✓ Area Optimization
- ✓ Higher Throughput

Meanwhile, this design reduces power consumption to some extent, for the power consumption is directly related to the chip area.

A implementation of area-optimized AES algorithm which meets the actual application is proposed in this thesis. After being coded with Verilog Hardware Description Language, the waveform simulation of the new algorithm was taken in the Xilinx 9.2 family. Ultimately, a synthesis simulation of the new algorithm has been done. The result shows that the design with the pipelining technology and special data transmission mode can optimize the chip area effectively. Meanwhile, this design reduces power consumption to some extent, for the power consumption is directly related to the chip area. Therefore the encryption device implemented in this method can meet some practical applications.

As the S-box is implemented by look-up-table in this design, the chip area and power can still be optimized. So the future work should focus on the implementation mode of S-box.
Mathematics in Galois field (28) can accomplish the bytes substitution of the AES algorithm, which could be another idea of further research.

Future Scope

The result shows that the design with the pipelining technology and special data transmission mode can optimize the chip area effectively. Therefore the encryption device implemented in this method can meet some practical Applications like image encryption.

In this thesis, we have studied AES encryption and decryption schemes and have highlighted some of the important mathematical properties as well as the security issues of AES algorithm. Since AES provides better security and has less implementation complexity, it has emerged as one of the strongest and most efficient algorithms in existence today. Hence, the optimal solution is the use of a hybrid encryption system in which typically AES is used to encrypt large data block. The future work can done for the distribution of secret key that is considered as a critical issue of AES like other symmetric encryption algorithm.

REFERENCES

[1] AI-WEN LUO, QING-MING YI, MIN SHI "Design and Implementation of Area-optimized AES Based on FPGA" Published by 2011 International Conference on Business Management and Electronic Information.

[2] Ahmed, S.; Samsudin, K.; Ramli, A.R.; Rokhani, F.Z. "Effective implementation of AES-XTS on FPGA" Published in TENCON 2011 - 2011 IEEE Region 10 Conference

.

[3] Kuo-Huang Chang[1], Yi-Cheng Chen[2], Chung-Cheng Hsieh[1], Chi-Wu Huang[2] and Chi-Jeng Chang[1] "Embedded a low area 32-bit AES for image encryption/decryption application" Published by IEEE International Symposium on Circuits and Systems, 2009. ISCAS 2009

[4] Shanxin Qu; GuochuShou; Yihong Hu; ZhigangGuo; Zongjue Qian "High Throughput, Pipelined Implementation of AES on FPGA" Published in International Symposium on Information Engineering and Electronic Commerce, 2009. IEEC '09.

[5] Kaur, Swinder; Vig, Renu**,** "Efficient Implementation of AES Algorithm in FPGA Device"Published in International Conference on Conference on Computational Intelligence and Multimedia Applications 2007.

[6] Helion Technologies Ltd, "Fast AES XTS/CBC Core for Xilinx FPGA" – (XEX-based Tweaked Codebook with Ciphertext Stealing), IP Core,http://www.heliontech.com/aes_xex.htm.

[7] Hatzidimitriou, E.; Kakarountas, A.P.; , "Implementation of a P1619 crypto-core for Shared Storage Media," MELECON 15thIEEE Mediterranean Electrotechnical Conference , vol., no., pp.597-601,

[8] M. Dworkin, Recommendation for Block Cipher Modes of Operation: The XTS-AES Mode

for Confidentiality on Storage Devices, NIST Special Publication 800-38E, US Nat'l Inst. of Standards and Technology, http://csrc.nist.gov/publications/nistpubs/800 38E/nist-sp-800-38E.pdf.

[9] Martin, L.; "XTS: A Mode of AES for Encrypting Hard Disks," Security & Privacy, IEEE , vol.8, no.3, pp.68-69.