ISSN 2319 – 2518 www.ijeetc.com
Vol. 3, No. 3, July 2014
© 2014 IJEETC. All Rights Reserved

Research Paper

DESIGN AND IMPLEMENTATION OF ADVANCED ENCRYPTION ALGORITHM WITH FPGA AND ASIC

Iyli Sagar[1]* and U Eranna[1]

*Corresponding Author: Iyli Sagar, sagar.iyli@gmail.com

A public domain encryption standard is subject to continuous, vigilant, expert cryptanalysis. AES is a symmetric encryption algorithm processing data in block of 128 bits. Under the influence of a key, a 128 bit block is encrypted by transforming it in a unique way into a new block of the same size. To implement AES Rijndael algorithm on FPGA using Verilog and synthesis using Xilinx, Plain text of 128 bit data is considered for encryption using Rijndael algorithm utilizing key. This encryption method is versatile used for military applications. The same key is used for decryption to recover the original 128 bit plain text. For high speed applications, the Non LUT based implementation of AES S-box and inverse S-box is preferred. Development of physical design of AES-128 bit is done using cadence SoC encounter. Performance evaluation of the physical design with respect to area, power, and time has been done. The core consumes 10.21 mW of power for the core area of 332128.742 μm^2.

Keywords: Encryption, Decryption, FPGA implementation

INTRODUCTION

The Rijndael proposal for AES defined a cipher in which the block length and the key length can be independently specified to be 128, 192, or 256 bits. The AES specification uses the same three key size alternatives but limits the block length to 128 bits. A number of AES parameters depend on the key length as shown in Table 1.

Table 1: AES Parameters

Key Size (Words/ Bytes/Bits)	4/16/128	6/24/192	8/32/256
Plaintext Block Size (Words/Bytes/Bits)	4/16/128	4/16/128	4/16/128
Number of Rounds	10	12	14
Round Key Size (Words/Bytes/Bits)	4/16/128	4/16/128	4/16/128
Expanded Key Size (Words/Bytes)	44/176	52/208	60/240

RELATED WORK

An overview of crypt analysis research for the advanced encryption standard is explained by Alan Kaminsky *et al.* Linear Cryptanalysis

[1] BITM, E&C Department, Bellary.

exploits approximate linear relationships that exist between inputs and outputs of a function block (Matsui, 1994). In the case of a block Cipher, linear combinations of plaintext patterns and linear combinations of Cipher text patterns are compared to linear combinations of key bits. Differential cryptanalysis exploits relationships that exist between differences in the input and output of a function block (Ben-Aroya and Biham, 1994). The advantages of a software implementation include ease of use, ease of upgrade, portability, and flexibility. However, a software implementation offers only limited physical security, especially with respect to key storage (Doud, 1999). Conversely, cryptographic algorithms (and their associated keys) that are implemented in hardware are, by nature, more physically secure, as they cannot easily be read or modified by an outside attacker (Doud, 1999). AES that was implemented in HDL, led to the use of a bottom-up design and test methodology (Phillips and Hodor, 1996). Arithmetic operations, architectural requirements, scalability and cost are widely considered in Riaz and Heys (1999) and Elbirt (1999). Shuenn-Shyang Wang and Wan-Sheng Ni (2004) proposes an efficient FPGA implementation of AES.

The hardware-based implementation of AES Rijndael algorithm (Daemen and Rijmen, 1999) is required because it can be more secure and consumes less power than software implementation. Standaert (2003) proposes a methodology to efficiently implement block cipher within commercially available FPGA and it is applied to design AES Rijndael which is shown to improve previously reported results in terms of hardware cost, throughput or efficiency. Mcloone and Mccanny (2001) presented a FPGA encryptor design that utilizes look-up table to implement the entire AES Rijndael round function. This is a more efficient FPGA implementation of AES Rijndael algorithm. Every component of AES algorithm is optimally designed to reduce the critical path and chip area.

Zhang and Parhi (2004) have suggested new implementation of multiplicative inverse in GF(2^4) for realizing S-box and inverse S-box using arithmetic in GF($(2^4)^2$) which is further decomposed into GF($((2^2)^2)^2$). Zhang and Parhi (2004) have also described efficient MixColumns and Inverse MixColumns implementation and evaluated the complete architecture for realizing fully unrolled AES implementation on FPGAs using 'on the fly' pipelined key schedule. Liu and Parhi (2008) proposed pre-computation techniques using which some computation in the critical path is eliminated to further reduce the critical path of the composite field arithmetic based S-box.

Choosing the suitable value for the subfield reducing polynomial was investigated extensively by O'driscoll. through a method which computes all possible Cyclotomic Cosets over 2 of degree 4 in GF(2^8) he concluded that there were only three choices for a reducing polynomial in the subfield GF(2^4) (Pong Chu, 2008). Most implementations conventionally make use of the memory intensive look up table approach for Substitute Byte/Inverse Substitute Byte block implementations resulting in an unbreakable delay (Nalini *et al.*, 2009).

PROBLEM DEFINITION

Implementation of AES- IP Core for 128 bit by employing a memory less combinational design for the generation of S-box and Inverse S-box. As an alternative to achieve higher speeds by eliminating memory access delay while reducing overall area occupied and power consumed by the AES core.

DESIGN METHODOLOGY OF AES ALGORITHM

The two basic hardware design methodologies currently available are Language-Based (High-Level) design and Schematic Based (Low-Level) design. Language-Based design relies upon synthesis tools to implement the desired hardware. While synthesis tools continue to improve, they rarely achieve the most optimized implementation in terms of both area and speed when compared to a schematic implementation. As a result, synthesized designs tend to be (slightly) larger and slower than their schematic based counterparts. Additionally, implementation results can greatly vary depending on the synthesis tool as well as the design being synthesized.

Block Diagram - Encryption

The encryption component processes plain text – 128 bit, round key schedule and produces the encrypted cipher text – 128 bit depending on the control signals as follows, i.e., In Encryption mode of operation load will be forced to 1 to load the plain text - 128 bit and key schedule – 128 bit. Reset will be made 1, after loading the data, load will be forced to zero to process the internal blocks, Subbytes, shift rows, and mix column after 10 rounds of process, done signal goes to high and encrypted output is available in the output register.

Decryption

The decryption component process encrypted data (cipher text – 128 bit) and encrypted key – 128 bit which has obtained in the 10th round

Figure 1: AES Encryption Block Diagram

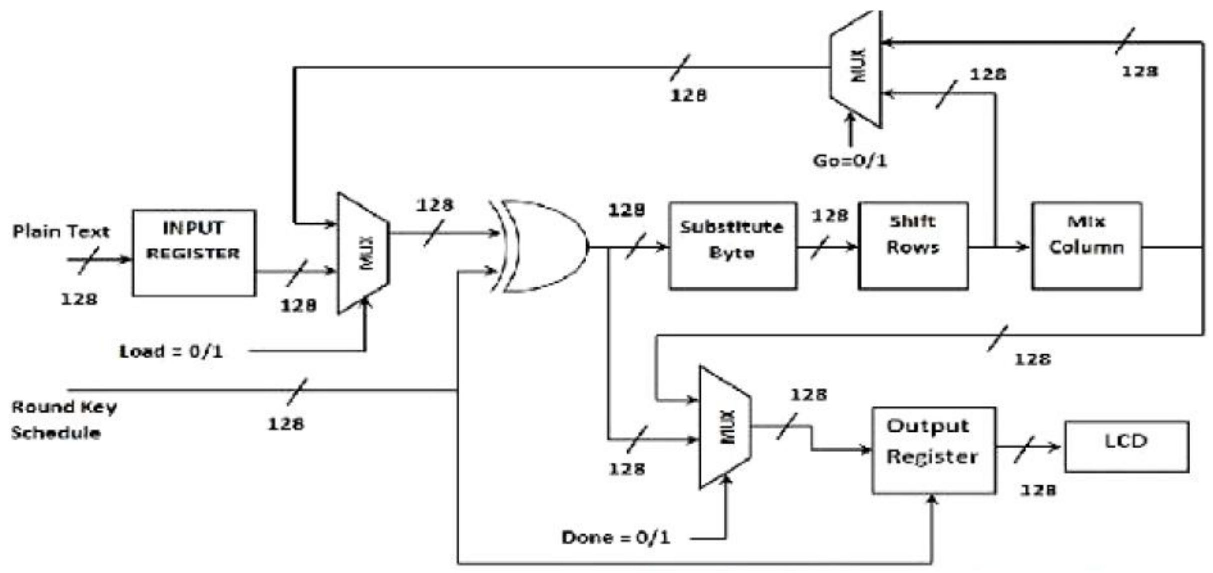
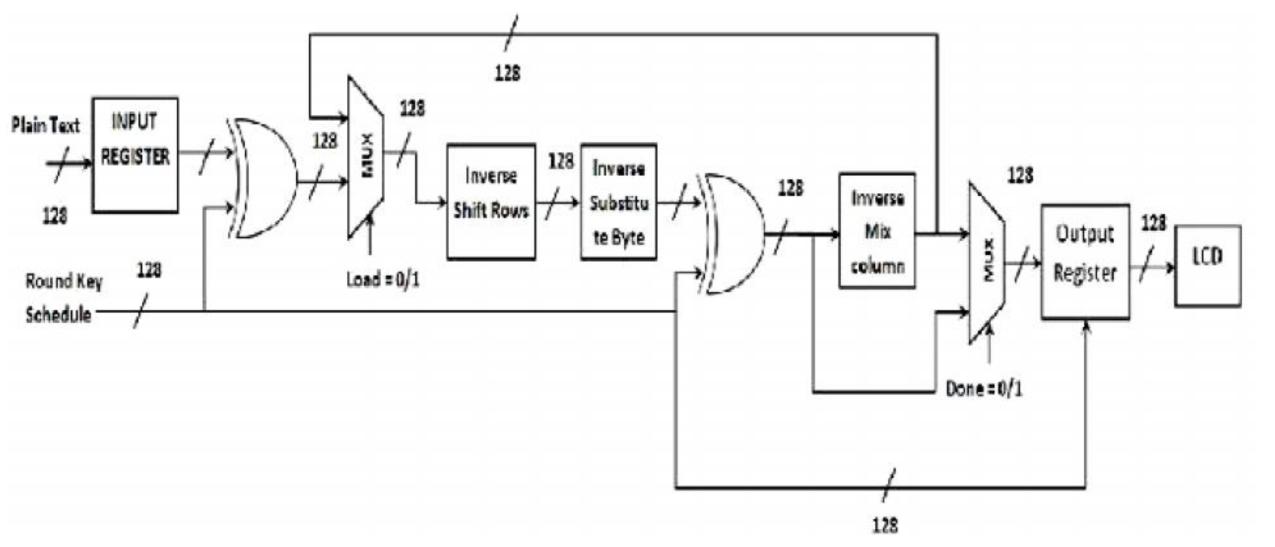


Figure 2: AES Decryption Block Diagram



operation of encryption and produces the decrypted output (plain text – 128 bit) and the key – 128 bit (which has been used in the initial round operation of encryption process). Depending upon the control signals, In decryption mode of operation load will be forced to 1 to load Encrypted input data 128 bit and key schedule 128 bit and reset is made as 1 after loading the inputs load is made to zero and internal blocks inverse shift rows, inverse substitute byte and inverse mix columns are processed after 10 round of operation, done signal will goes to high and decrypted output will be available in the output register.

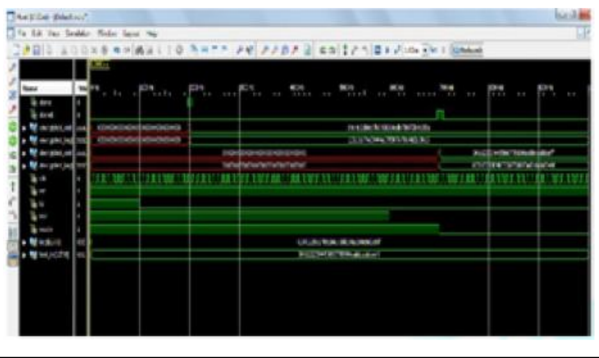
VERILOG IMPLEMENTATION

The Verilog HDL is an IEEE standard hardware description language. It is widely used in the design of digital integrated circuits. Verilog is intended to be used for verification through simulation, for timing analysis, for test analysis and for logic synthesis (Douglas Smith, 1996). Verilog HDL allows designers to design at various levels of abstraction like register transfer level, gate level and switch level. Verilog is used as an input for synthesis programs which will generate a gate-level description for the circuit. Xilinx ISE 13.2 is a software tool developed by Xilinx for synthesis and analysis of HDL designs.

SIMULATION RESULT USING MODEL SI

Figure 3 shows the simulation results using ModelSim for AES-128 bit by considering the following inputs. Plain Text – 00112233445566 778899aabbccddeeff Key Input –

Figure 3: Simulation Result for Both Encryption and Decryption



000102030405060708090a0b0c0d0e0f. The following Encrypted output obtained is as shown.

FPGA IMPLEMENTATION OF THE DESIGN

Figure 4 shows the RTL schematic of AES-128 bit which comprises of AES Encryption and Decryption.

Figure 4: RTL Schematic of AES-128 bit

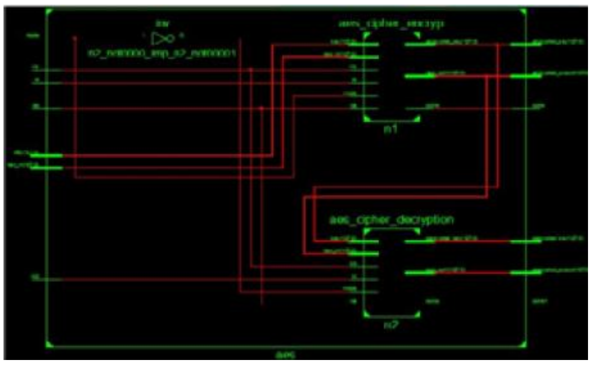
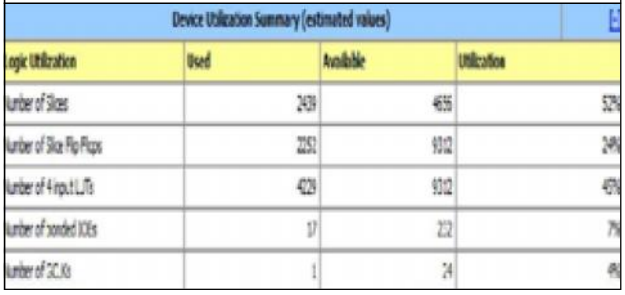
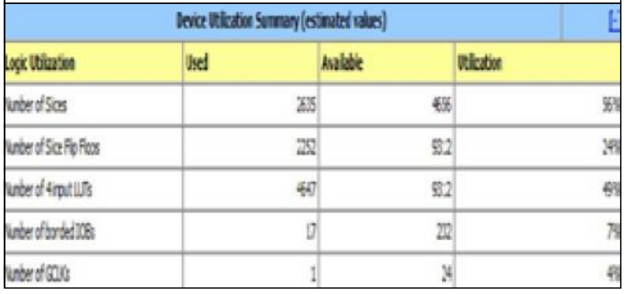


Table 2: Device Utilization Summary of AES Encryption



Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slices	2439	4656	52%
Number of Slice Flip Flops	2252	9312	24%
Number of 4 input LUTs	4229	9312	45%
Number of bonded IOBs	17	232	7%
Number of GCLKs	1	24	4%

Table 3: Device Utilization Summary of AES Decryption



Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slices	2635	4656	56%
Number of Slice Flip Flops	2252	9312	24%
Number of 4 input LUTs	4647	9312	49%
Number of bonded IOBs	17	232	7%
Number of GCLKs	1	24	4%

Table 2 Shows the Device Utilization Summary of AES Encryption it comprises of Add Round Key schedule Substitute Byte, Mix Column.

Table 3 above Shows the Device Utilization Summary of AES Decryption it comprises of Add Round Key Schedule Inverse-Substitute Byte, Inverse-Mix Column.

HARDWARE RESULTS

The Figure 5 shows the recovered plain text and original key displayed concidering the following encrypted inputs to the decrypter.

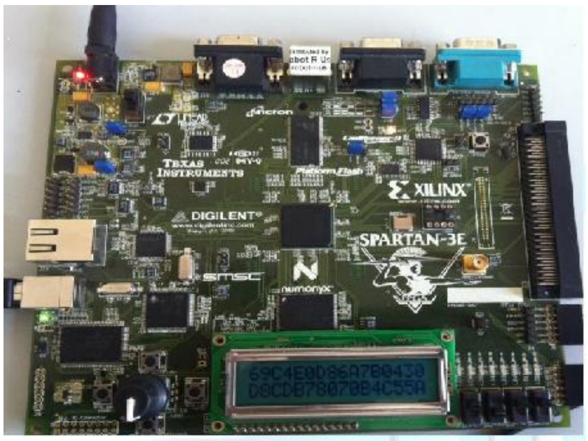
Encrypted Input to the decrypter –

69c4e0d86a7b0430d8cdb78070b4c55a

Input key to the decrypter –

13111d7fe3944a17f307a78b4d2b30c5

Figure 5: Snap Shot of Cipher Key, Plain Text, Recovered Plain Text, Recovery Key on FPGA



ASIC IMPLEMENTATION OF THE DESIGN

The designs AES is synthesized using Cadence RTL compiler and the steps involved in the synthesis of the design are similar to the logic synthesis in Xilinx.

Table 4: Area Report of AES Core in Physical Design

Total area of standard cells	331948.109 um^2
Total area of standard cells (subtracting physical cells)	233473.363 um^2
Total area of macros	0.000 um^2
Total area of core	332128.742 um^2
Total area of chip	381630.419 um^2

PHYSICAL DESIGN OF AES 128 BIT

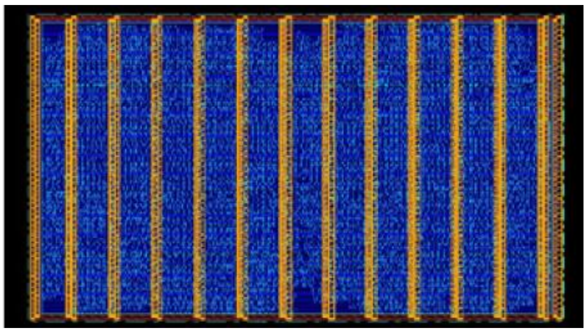
The first step in the physical design flow is floor planning. Followed by Partitioning, which is a process of dividing the chip into small blocks. After partitioning, Placement is performed in four optimization phases:

1. Pre-Placement optimization
2. In placement optimization
3. Post Placement Optimization (PPO) before Clock Tree Synthesis (CTS)
4. PPO after CTS

The goal of Clock Tree Synthesis (CTS) is to minimize skew and insertion delay. After Routing Physical verification checks the correctness of the layout design. Once the design has been physically verified, optical-lithography masks are generated for manufacturing. The layout is represented in the GDSII stream format that is sent to a semiconductor fabrication plant.

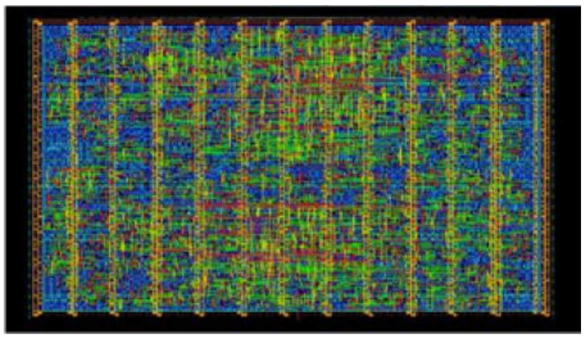
As described above the first step in physical design is importing the design files and libraries, then floor plan, where the area allocated for the design on the chip will be decided and the power planning for the design will be done. The next step is special routing to provide power to the standard cells. The next step is the placement of standard cells.

Figure 6: Physical Layout of a AES After Standard Cell Placement



The design is further checked for DRC, ARC and ERC and the GDSII file has been extracted. The Figure 7 shows the complete physical layout of the AES core after Timing Analysis and Routing.

Figure 7: Physical Layout of AES Core



Throughput Calculation

Maximum output required time after clock (T) = 4.496 ns

$$\text{Frequency (F)} = \frac{1}{T} = \frac{1}{4.496\ ns} = 222.41\ \text{MHz}.$$

Throughput for 128 bit AES = 2.84 Gbps.

COMPARISON RESULTS

From the above comparisons, optimized design achieves a throughput of 0.72855 Gbps which is faster compared to the previous

Table 5: FPGA Comparison Table for Frequency and Throughput

Design	Device	Fmax (Mhz)	Through Put	Slices (Gbps)	Mbps/ Slice
Elbirt *el al.**	XCV1000-4	31.8	0.4070	10992	0.0376
Monica liberatori *et al.*#	XC3S500E-4FG320	9.375	0.120	1643	0.0730
Ours	XC3S500E-4FG320	222.41	2.846	2439	1.166

Table 6: ASIC Comparison Table for Area and Power

Design	Area	Power
Sumanth Kumar Reddy S *et al.*$	2258469 μm^2	655.5 mW
Ours	332128 μm^2	10.21 mW

FPGA implementations. The area occupied by the AES Core is 332128 μm^2 and power consumed by the core is 10.21 mW.

CONCLUSION AND FUTURE WORK

The designed core supports both encryption and decryption standards. Its functionality has been verified using simulation, by taking various inputs and is synthesized by using Xilinx 13.2. The design is targeted on FPGA (spartan-3E).The design uses 2439 slices, 2252 flip flops, 4647 4-input look up tables and operates at 2.84 Gbps (Throughput). Development of physical design of AES-128 bit is done using cadence SoC encounter. Performance evaluation of the physical design with respect to area, power, and time has been done. The core consumes 10.21 mW of power for the core area of 332128.742 μm^2.

This Implementation of 128 bit AES using Rijndael algorithm, and the same can be extended to encrypt 192 and 256 bits of plain text data with proper key length, which makes even tougher to decrypt the original data form an unauthorized receivers.

REFERENCES

1. Alan Kaminsky, Michael Kurdziel and Stanisaw Radziszowski (2010), "An Overview of Cryptanalysis Research for the Advanced Encryption Standard", IEEE, the 2010 Military Communications Conference – Unclassified Program-Cyber Security and Network Management, p. 1310.

2. Ben-Aroya I and Biham E (1994), "Differential Cryptanalysis of Lucifer", *Crypto, Journal of Cryptology*, pp. 187-19l, Springer.

3. Daemenand J and Rijmen V (1999), "AES Submission Document on Rijndael", Version 2, September (http://csrc.nist.gov/cryptotoolkit/AES/Rijndael/Rij ndael.pdf).

4. Doud R (1999), "Hardware Crypto Solutions Boost VPN", *Electron. Eng. Times*, April 12, pp. 57-64.

5. Douglas J Smith (1996), "HDL Chip Design Using VHDL or Verilog", Doone Publications.

6. Elbirt (1999), "An FPGA Implementation and Performance Evaluation of the Cast-256 Block Cipher", May, Cryptography and Information Security Group, ECE Department, Worcester Polytechnic Institute, Worcester, Ma, Tech. Rep.

7. Liu R and Parhi K K (2008), "Fast Composite Field Architectures for Advanced Encryption Standard", Proceedings GLSVLSI'08, May 4-6, pp. 65-70, Orlando, Florida, USA.

8. Matsui M (1994), "Linear Cryptanalysis Method for DES Cipher", *Eurocrupt, Lncs*, Vol. 765, pp. 386-397, Springer.

9. Mcloone W and Mccanny J V (2001), "Rijndael FPGA Implementation Utilizing Look-Up Tables Signal Processing Systems", *2001 IEEE Workshop on Signal Processing Systems*, pp. 349-360.

10. Menezes A, Van Oorschot P and Vanstone S (1997), "Handbook of Applied Cryptography", pp. 81-83, CRC Press, New York.

11. Monicalibertori, Fernando Otero, Bonadero J C and Jorge Castineira (2007), "AES-128 Cipher High Speed, Low Cost FPGA Implementation", *IEEE*, April, pp. 195-198.

12. Nalini C, Anandmohan P V, Poonaih D V and Kulkarni V D (2009), "Compact Designs of Sub Bytes and Mix Column for AES", IEEE International Advance Computing Conference (IACC), March, pp. 1241-1247.

13. O'Driscoll C (2001), "Hardware Implementation Aspects of the Rijndael Block Cipher", Masters Thesis, National University of Ireland, Cork, Ireland.

14. Phillips C and Hodor K (1996), "Breaking the 10 k FPGA Barrier Calls for an ASIC-Like Design Style", *Integrated Syst. Design*.

15. Pong P Chu (2008), "FPGA Prototyping by Verilog Examples", Wiley Publications.

16. Riaz M and Heys H (1999), "The FPGA Implementation of RC6 and Cast-256 Encryption Algorithms", in Proc. IEEE 1999 CAN. Conf. Electrical and Computer Engineering, March, Edmonton, Alta., Canada.

17. Rudra A, Dubey P K, Jutla C S, Kumar V, Rao J R and Rohatgi P (2001), "Efficient Implementation of Rijndael Encryption with Composite Field Arithmetic", in Proceedings of the Cryptographic Hardware and Embedded Systems Conference, Lecture Notes in Computer Science, Vol. 2162, pp. 171-185, Paris, France.

18. Sheunn-Shyang Wang and Wan-Sheng Ni (2004), "An Efficient FPGA Implementation of Advanced Encryption Standard Algorithm", *International Symposium on Circuits and Systems, IEEE*, p. 597.

19. Standaert F X (2003), "AMethodology to Implement Block Ciphers in Reconfigurable Hardware and as Application to Fast and Compact AES Rijndael", The Field Programmable Logic Array Conference, pp. 216-224, Monterey, California.

20. Sumanth Kumar Reddy S, Sakthivel R and Praneeth P (2011), *International Journal of Advanced Engineering Sciences and Technologies (IJAEST)*, Vol. 6, No. 1, pp.022-026.

21. Wong M M and Wong M L D (2010), "A High Throughput Low Power Compact

AES s-Box Implementation Using Composite Field Arithmetic and Algebraic Form Representation", Proc. IEEE 2nd Asia Symposium on Quality Electronic Design, pp. 318-323.

22. Zhang X and Parhi K K (2004), "High Speed VLSI Architectures for AES Algorithm", *IEEE Transactions on VLSI Systems*, Vol. 12, No. 9, pp. 957-967.