International Journal of Advance Research in Science and Engineering
Volume No.06, Special Issue No.(01), December 2017
www.ijarse.com
IJARSE
ISSN: 2319-8354

Efficient Advance Encryption Standard (AES) Implementation On FPGA Using Xilinx System Generator

B SATYA NARAYANA[1], DR. R.P. SINGH[2]

[1,2](Research Scholar, Sri Satya Sai University of Technology and Medical Sciences, India)

ABSTRACT

We propose an efficient hardware architecture design & implementation of Advanced Encryption Standard (AES). The cryptographic algorithms can be implemented with software or built with pure hardware. However Field Programmable Gate Arrays (FPGA) implementation offers quicker solution and can be easily upgraded to incorporate any protocol changes. This contribution investigates the AES encryption cryptosystem with regard to FPGA and Very High Speed Integrated Circuit Hardware Description language (VHDL). Optimized and Synthesizable VHDL code is developed for the implementation of 128-bit data encryption process. AES encryption is designed and implemented in FPGA, which is shown to be more efficient than published approaches. Xilinx ISE 12.3i software is used for simulation. Each program is tested with some of the sample vectors provided by NIST and output results are perfect with minimal delay. The throughput reaches the value of 1609Mbit/sec for encryption process with Device XC6vlx240t of Xilinx Virtex Family.

I.INTRODUCTION

In the era of this modern world, secrecy and security are the major concerns which should be kept in mind with respect to digital computer systems. Cryptography provides solutions regarding fool proof secrecy, security, and reliability of given information. It is keeping its vital function in different applications which includes online banking system, Cellular networks, computer hardware emulations, medical imaging, software defined radios, bioinformatics and wireless communication etc. Reconfigurable platform like FPGA are the best for implementation of cryptographic algorithms [1]. These platforms are reconfigurable to provide time and cost effective solutions as compared to Application Specific Integrated Circuit (ASIC) [2]. A reconfigurable platform provides improved performance than software implementations and can also be reconfigured on the fly to store the updated encryption standard.

In recent years, digital hardware design seems to be more similar to the software design, driven by the increased complex design, time-to-market anxiety and demand for an effective participation between various project teams. Platform-based design has become appropriate for IC design projects in this digital world. To minimize the algorithmic process time in term of plenty of data, it is very much inevitable to adopt and implement the algorithm of hardware, despite the fact that software implementation can only meet the requirement of low cost for users. In order to attain a balance between the cost and time, an efficient method must be explored and

implement for various combinations of hardware and software to realize algorithmic best solutions of different requisite.

In this work we focus on the efficient and well organized implementation of AES architecture on FPGA using HLL approach i.e. Xilinx System Generator. The proposed FPGA platform for the implementation of this work is Virtex-5 FPGA from Xilinx. HLLs provide a better way to achieve the high-performance in reconfigurable computing to implement any design in hardware [3]. It provides a great deal of functional abstraction and develops highly parallel systems. High level language tool automatically maps the model design to efficient hardware implementation.

II.ADVANCED ENCRYPTION STANDARD

The National Institute of Standards and Technology (NIST) announced that Rajndael pronounced as "Rain Doll"planned by two Belgium researchers Joan Daemen and Vincent Rijment was adopted as Advanced Encryption Standard (AES) for encryption and decryption of blocks of data. The draft is published in December 2001,under the name as FIPS-197(Federal Information Processing Standard number 197). The criteria defined by selecting AES fall into three areas Security, Implementation and cost of the algorithm. The main emphasis was the security of the algorithm to focus on resistance of cryptanalysis attacks, implementation cost should be less so it canbe used for small devices like smart cards. The AES algorithm is a private key block cipher. It encrypts data of block size 128bits. It uses key sizes, 128bits. AES uses three different types of round operations. One of the main features of AES is simplicity that is achieved by repeatedly combining substitution and permutation computations at different rounds. That is, AES encrypts/decrypts a 128-bit plaintext/cipher text by repeatedly applying the same round transformation a number of times depending on the key size. Advanced Encryption Standard (AES) algorithm not only for security but also for great speed. Advanced Encryption Standard not only assures security but also improves the performance in a variety of settings such as smartcards, hardware implementations etc.AES is federal information processing standard and there are currently no known non-brute-force direct attacks against AES.AES is strong enough to be certified for use by the US government for top secret information.

III.FEATURES OF AES ENCRYPTION ALGORITHM

- Advanced Encryption Standard (AES) algorithm works on the principle of Substitution Permutation network.
- AES doesn't use a Feistel network and is fast in both software and hardware.
- AES operates on a 4×4 matrix of bytes termed as a state
- The Advanced Encryption Standard cipher is specified as a number of repetitions of transformation sounds that convert the input plaintext into the final output of cipher text.
- Each round consists of several processing steps, including one that depends on the Encryption key.

International Journal of Advance Research in Science and Engineering
Volume No.06, Special Issue No.(01), December 2017
www.ijarse.com
IJARSE
ISSN: 2319-8354

The AES is an iterative algorithm and uses four operations in different rounds, namely Sub Bytes, Shift Rows, MixColumns and Key Additions transformations as shown in fig.1 Sub Bytes transformation is done through S-box. S-box is the vital component in the AES architecture that decides the speed/throughput of the AES.

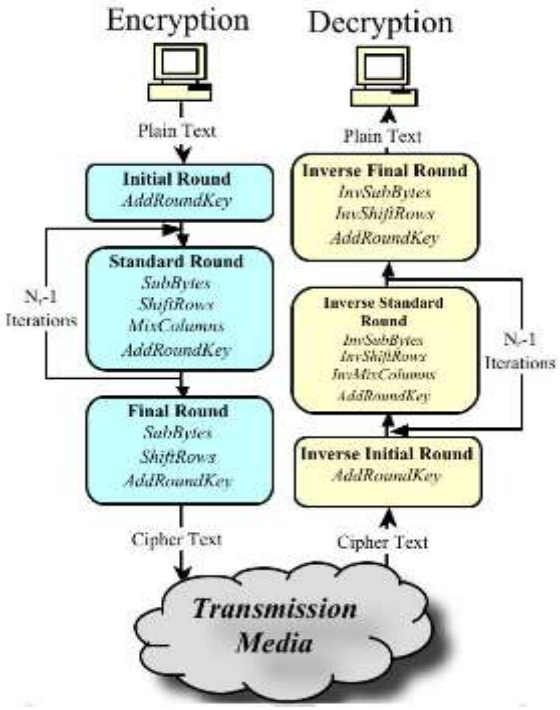

Fig. 1 AES Algorithm

IV.IMPLEMENTATION

Iterative and pipelined architectures are the two basic reconfigurable architectures that are commonly used for the implementation of encryption functions depending on type of application ranging from low to high speed. In this work we adopted pipelined architecture to implement AES encryption function in order to get best possible results in term of throughput. Further we have used the HLL approach to directly map our design on FPGA. In addition, our design is not device-specific; System Generator is highly scalable and can synthesis a design to different FPGA chips that leads to more flexible and fast design.

The block diagram of our full 128-bit AES pipelined architecture is shown in Fig 2, where each round is implemented separately in hardware by enclosing the four transformations as subsystems except in the last round where Mix-Column transform has been eliminated. Initial round consists of only Add Round Key transformation where input data is XORed with the initial Round Key value. Registers are placed at the end of each round forming hierarchical stages within each round of the algorithm. Each 128-bit transform; Sub Byte, Mix Column, Add Roundkey consists of four parallel copies of 32-bit modules. The detailed implementation and optimization of each is as follows:

International Journal of Advance Research in Science and Engineering
Volume No.06, Special Issue No.(01), December 2017
www.ijarse.com
IJARSE
ISSN: 2319-8354

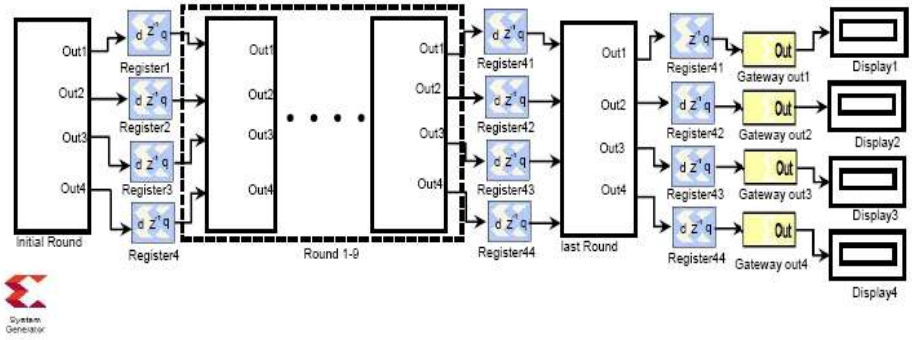

Fig 2. Pipelined architecture of AES in System Generator

V.SUB BYTE AND SHIFT ROW

There are two basic methods for generating SubByte of AES, either by using multiplicative inverse or by using memory based table lookup. We designed the SubByte using lookup based approach with the help of Dual-Port RAM to store 256 lookup values. Dual-Port RAM is configured as a ROM to access 8-bit lookup values corresponding to each 8-bit input addresses, for this we have operated the Dual-Port RAM in "no read on write" mode. Input of constant zero is given to the data input pin & write enable pin of RAM as we have not required here to write the data. Lookup values are directly stored in the dual port RAM with the help of coefficient file in the form of decimal numbers.

The input bytes are delivered to the address pins addra [7:0] and addrb [7:0] of the BRAM while corresponding lookup values will be taken from the output pins A [7:0] and B [7:0] of the Dual-Port RAM. Input of 128-bit State is arranged in the four 32-bits words and each is directly given to the 4 32-bit SubByte block. The details of our proposed 32-bits SubByte architecture is shown in Fig 3 which consists of two Dual-Port RAMs.

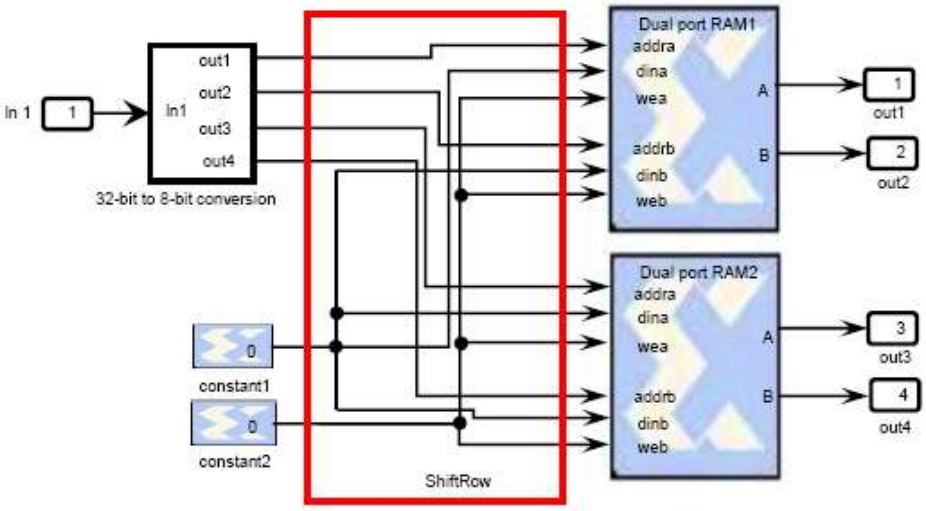

Fig 3. System Generator SubByte

International Journal of Advance Research in Science and Engineering
Volume No.06, Special Issue No.(01), December 2017
www.ijarse.com
IJARSE
ISSN: 2319-8354

Before applying the 32-bit data to each port of the Dual port RAMs for lookup, we first extract 8-bit data from 32-bit input. Because each RAM is able to lookup 8-bit data and we cannot apply the 32-bit data directly to Dual port RAMs. So for this we have used 32-bit to 8-bit conversion block. We can extract our desired 8 bit data by using the slice blocks. But this approach is not efficient in term of resource as it will require numerous slice resources. Therefore we redesigned this block by using custom logic in which we have used the right and left shift methodology to extract 8 bit data which helped us to save resources up to 1700 slices. The detail of 32-bit to 8-bit conversion module is shown in Fig 4. The Shift Row transformation is also implemented within the same SubByte block to save the resources. This is done simply by rearranging the wires according to the shifted data as marked by red box in Fig 3 and applied directly to the RAMs.

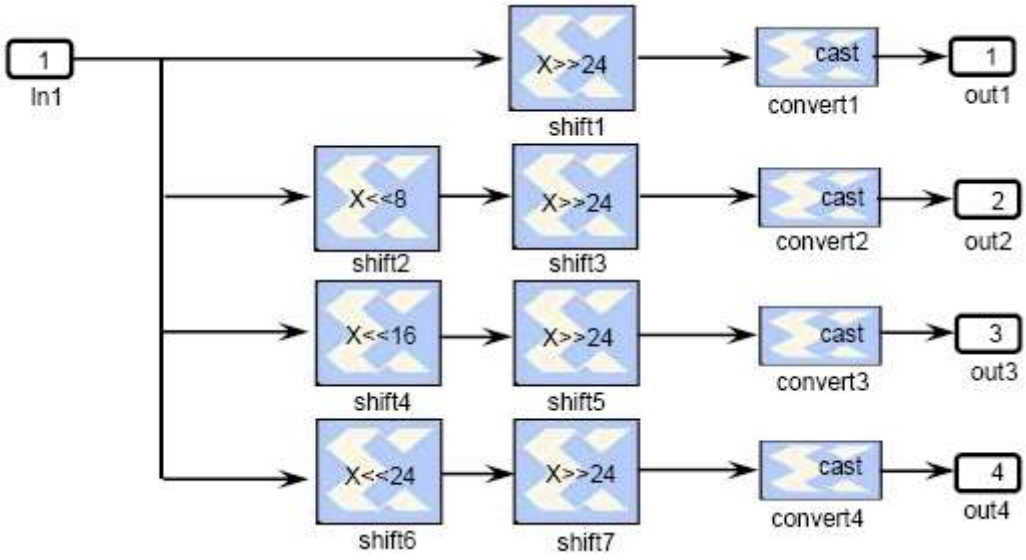

Fig 4: 32 bit to 8 bit Conversion

VI.MIXCOLUMN

In MixColumn block we need a multiplier of 1, 2 and 3. However the implementation of multipliers is very costly in hardware. MixColum use multiply by constant and it can efficiently implemented using multiplier less approach to save valuable hardware resources. We implement our multipliers by using shift and add algorithm to save the utilization cost. The architecture of MixColumn for first 32-bit data is shown in Fig 5, 32-bit data is passed directly through a connection wire when multiplied by number "1". Here in this block we have designed two multipliers to carry out the multiplication between input data and the numbers "2" and "3".

International Journal of Advance Research in Science and Engineering
Volume No.06, Special Issue No.(01), December 2017
www.ijarse.com
IJARSE
ISSN: 2319-8354

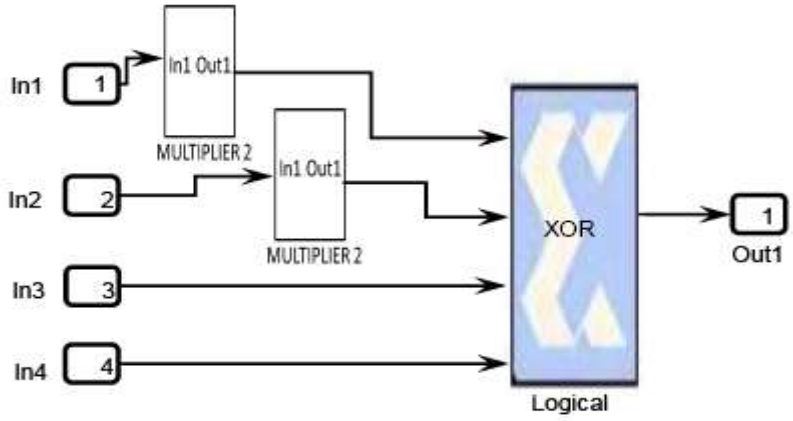

Fig 5: MixColumn

Multiplier 2 is designed by using shift block where left shift is applied to input data which results in multiplication by number "2" as shown in Fig 6.

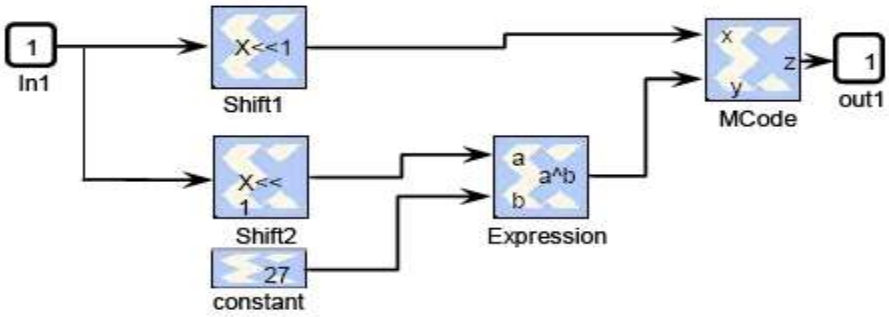

Fig 6: Multiplier 2

While Multiplier 3 is designed by dividing the number "3" into (2+1) where multiplication of data by number "2" is done by single left shift and the resultant number is then added to itself by using XOR operation as shown in Fig 7.

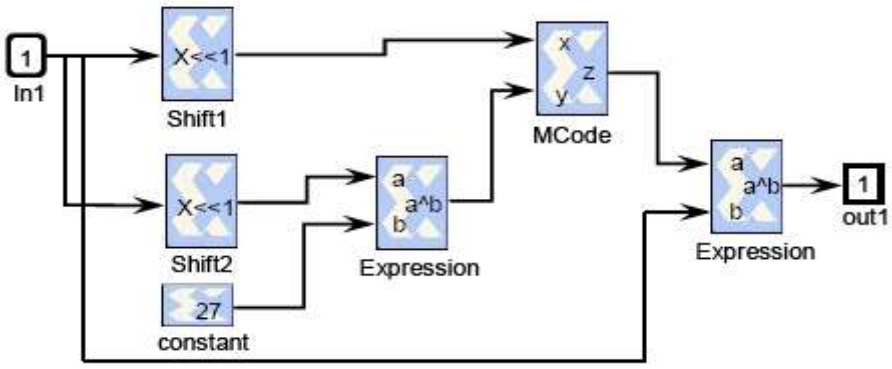

Fig 7: Multiplier 3

In both multipliers in order to satisfy the irreducible condition, the shifted data is XORed with the constant "27" (dec) in hex it is "1b" [2]. The mod 27 must be applied on the results of the multiplication in order to get the number within the range of 255. The MCode block is then designed to select the multiplication result "x" or "y", which is less than 255. It is programmed by the following code.

Function z = x1max(x, y)

if x < 256

z = x

VII.ADD ROUND KEY

Add Round Key transformation receives 16 bytes of data from Mix Column and performs XORing with the Round Keys. Round Keys are taken from FIPS-197 [2], converted to decimal numbers and are provided on the fly in the form of constants as shown in Figure 9. Here the expression blocks are used for XORing.

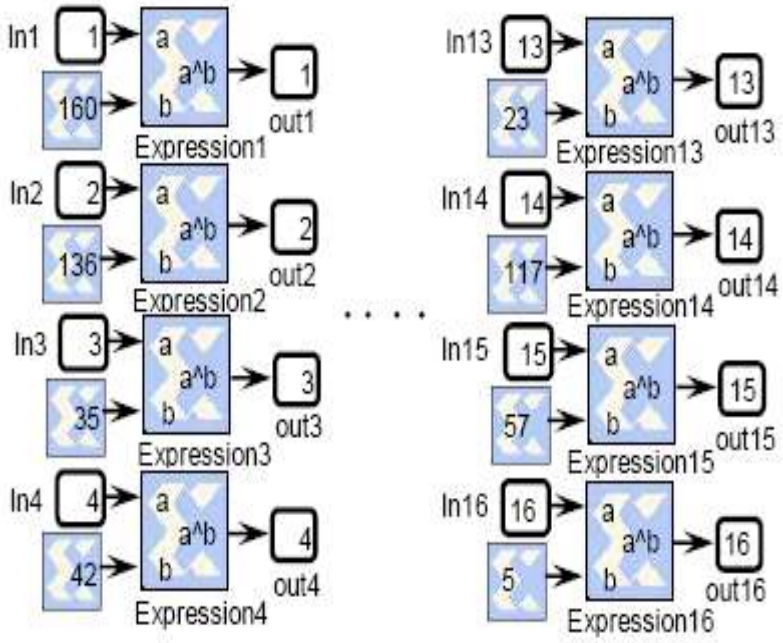

Fig 8: Add Round Key

VIII.VALIDATION

Result outputs and behavior of design is verified by giving the pre-defined test vectors defined in AES FIP-197 [15] against standard AES output for testing and validation. Each module is implemented and validated individually. The functionality of each step is tested and verified one by one. Then the verification of output of each round is performed and validated the results after combining all the rounds in the main module.

IX.RESULTS

Hardware implementation results are targeted for Xilinx Virtex-6 xc6vsx315t-3ff1156 FPGA. The design has been implemented using Xilinx System Generator tool in MATLAB to generate verilog code (.v file) alongwith test bench are finally synthesized and simulated using Xilinx ISE Foundation 13.1 and Mentor Graphic ModelSim, respectively. 128-bit AES pipeline architecture occupies 80 BRAM's for the implementation of SubByte and utilized 380 Slices for the remaining logic. The design operates on maximum frequency of 288.19MHz and offers high throughput of 36.864 Gbps as compared to previously reported designs.

Table 1 details the comparison results of previous AES implementations using HLL on different platforms. Parameters chosen for comparison are platform, to clearly indicate different high level implementation platforms, Data-path to indicate numbers of bit simultaneously processed by design. Operating frequency indicates maximum operational frequency of design and throughput to indicate performance of design.

It's evident from Table 1 that our performance results of proposed architecture gives better resource utilization and offers greater operating frequency as compared to all other previously reported high-level language tool implementations of AES. Mali et al [4] presents 128-bit AES implementation on Handle-C environment. This design operates at the maximum frequency of 74.4MHz and offers throughput of 7.76(Mbit/s). M. Askar et al. [5] presented 128-bit AES implementation using System C tool and System Crafter tool to translate the System C descriptions into hardware. It takes 153 machine cycles to process whole data and throughput of 90(Mbit/s). It reported 178 numbers of occupied slices for MixColumn and 197 number of slices for the implementation of ShiftRow, whereas our approach gives 380 slices for the whole architecture of AES. Osvik et al [9] previously reported 8-bit AES implementation on AVR and 32-bit AES. architecture on ARM platform that operates on the frequency of 124.6 cycles/byte and 34 cycles/byte respectively. Babu et al [10] implemented 128-bit architecture of AES using custom instruction method provided by ARM 7 with Keil platform. Hasamnis et al. [11] has given the high-level design architecture for 128-bit AES implementation where the algorithm is controlled through C-code written in NIOS II IDE that operates on frequency of 217.31020M Cycles. Next, Bos et al [12] reported the 128-bit AES implementation on NVIDIA graphics processing units (GPUs) and the Cell broadband engine. The GPU implementation delivers the throughput of 0.17 cycles per byte for encryption module and the Cell broadband engine offer the speed of 11.7 cycles per byte. In Biglari et al [13], the 128-bit pipeline architecture of AES is given on Maestro platform. This work presents tightly coupled encryption and round key generation modules in the main encryption module that enables the design to reach a throughput of 12.8 Gbps and runs at the speed of 100MHz. At the end, Mourad et al [14] presents the methodology which maps the AES design described in a high level language, Handel-C, to FPGA for low area consideration. This design reported 437 slices for encryption and offers the throughput of 1.716 Gbps.

Table 1: Comparison Results of AES

Implementation	Platform	Data-Path	Frequency	Throughput
Mali et al [4]	Handel-C	128	74.4MHz	7.76(Mbit/s)
Askar et al [5]	SystemC	128	153 Machine Cycles	90(Mbit/s)
Osvik et al [9]	AVR	8	124.6 cycles/byte	-
Osvik et al [9]	ARM	32	34 cycles/byte	-
Babu et al [10]	ARM	128	-	-
Hasamnis et al [11]	NIOS II IDE	128	217.31020M Cycles	-
Bos et al [12]	NVIDIA (GPUs)	128	-	0.17 cycles/byte
Bos et al [12]	Cell broadband engine	128	11.7 cycles/byte	-
Biglari et al [13]	Maestro	128	100 MHz	12.8 Gbps
Mourad et al [14]	Handel-C	128	-	1.716 Gbps
Our Implementation AES-128	System Generator	32	288.19MHz	36.864 Gbps

X.CONCLUSION

The paper presents reconfigurable platform used with High-level language approach and the work presented here uses efficient implementation of AES using Xilinx System Generator; the approach not only reduces the overall utilization but also gives good enough clock frequency and latency. This paper shows performance comparison to the various FPGA (Verilog) implementations. It is the user friendly design for HLL users and gives fast design to market.

REFERENCES

[1] N. A. Saqib, C. K. Koc, A .D. Pérez, F. Rodriguez-Henriquez, “Cryptographic Algorithms on Reconfigurable Hardware”, Signals and Communication Technology, Springer, vol. 26, pp. 362, 2007.

[2] M. Mozaffari-Kermani, A. Reyhani-Masoleh, "Efficient and High-Performance Parallel Hardware Architectures for the AES-GCM," IEEE Transactions on Computers, vol. 61, no. 8, pp. 1165-1178, Aug. 2012.

[3] E. El-Araby, Saumil G. Merchant, T. El-Ghazawi, "A Framework for Evaluating High-Level Design Methodologies for High-Performance Reconfigurable Computers," IEEE Transactions on Parallel and Distributed Systems, vol. 22, no. 1, pp. 33-45, Jan. 2011.

[4] M. Mali, F. Novak and A. Biasizzo, “Hardware Implementation of AES Algorithm”, Journal of Electrical Engineering, vol. 56, pp 265–269, 2005.

[5] M. Askar and T. Egemen, “Design and SystemC Implementation of a Crypto Processor for AES and DES Algorithms”, Information Security and Cryptology Conference with International Participation, Dec 2007.

[6] M. Lukowiak, S. Radziszowski and J. Vallino and C. Wood, “Cybersecurity Education: Bridging the Gap Between Hardware and Software Domains”.

[7] F. Oboril, I. Sagar, M. B. Tahoori, "A-SOFT-AES: Self-adaptive software-implemented fault-tolerance for AES", On-Line Testing Symposium (IOLTS), IEEE 19th International, pp.104-109, 8-10 July 2013.

[8] Y. Wang, Y. Ha, "FPGA-Based 40.9-Gbits/s Masked AES with Area Optimization for Storage Area Network", Circuits and Systems II: Express Briefs, IEEE Transactions, vol.60, no.1, pp.36-40, Jan 2013.

[9] D. Osvik, J. Bos, D. Stefan and D. Canright, “Fast Software AES Encryption”, FSE'10 Proceedings of 17th International Conference on Fast Software Encryption, pp 75-93, 2010.

[10] T. Babu, K. Murthy and G. Sunil, “AES Algorithm Implementation using ARM Processor”, 2nd International Conference and workshop on Emerging Trends in Technology (ICWET) Proceedings published by International Journal of Computer Applications (IJCA), 2011

[11] M. Hasamnis, P. Jambhulkar and S. Limaye, “Implementation of AES as a Custom”, Advanced Computing: An International Journal (ACIJ), vol.3, No.4, July 2012.

[12] J. Bos, D. Osvik, D. Stefan, “Fast Implementations of AES on Various Platforms”, IACR Cryptology ePrint Archive, vol. 501, 2009.

[13] M. Biglari, E. Qasemi, B. Pourmohseni, "Maestro: A high performance AES encryption/decryption system", Computer Architecture and Digital Systems (CADS), 17th CSI International Symposium, pp.145-148, 30-31 Oct. 2013.

[14] O. Mourad, S. Lotfy, M. Noureddine, B. Ahmed, T. Camel, "AES Embedded Hardware Implementation", Adaptive Hardware and Systems, Second NASA/ESA Conference, pp.103-109, 5-8 Aug. 2007.

[15] FIPS-197, “Federal Information Processing Standards Publication FIPS-197, Advanced Encryption Standard (AES)”, http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf, October 1999.

[16] N.Anitha Christy and P.Karthigaikumar, “FPGA Implementation of AES Algorithm using Composite Field Arithmetic”, International Conference on Devices, Circuits and Systems (ICDCS), pp. 713-717, 2012.

[17] A. C. Zigiotto, R. d'Amore, "A Low-Cost FPGA Implementation of the Advanced Encryption Standard Algorithm," 15th Symposium on Integrated Circuits and Systems Design, pp.191, 2002.

[18] S. Qu, G. Shou, Y. Hu, Z. Guo and Z. Qian, “High Throughput Pipelined Implementation of AES on FPGA”, International Symposium on Information Engineering and Electronic Commerce, pp. 542-545, 2009.

[19] V. Elamaran and G. Rajkumar, “FPGA Implementation of Point Processes Using Xilinx System Generator”, Journal of Theoretical and Applied Information Technology, vol. 41,pp. 201-206, July 2012.