AN EFFICIENT VLSI DESIGN OF AES CRYPTOGRAPHY BASED ON DNA TRNG DESIGN

Vikas J[1], Sowmya Sunkara[2]

[1]*MTech. in VLSI Design and Embedded Systems, BMSCE, Bangalore.*
[2]*Asst. Professor, Dept. of ECE, BMSCE, Bangalore, Karnataka, India.*

Abstract - *This paper shows the design of Advanced Encryption Standard (AES), focused on improving the security and reducing the area and delay. In AES-128 there will be 10 round of operations which requires 10 different set of keys that are generated using process called key expansion. The key expansion used is a manual process and also very complicated which occupies more area and delay. In embedded design, the speed and area are the major factors. The Key material which is used in cryptography is critical, the security of the entire framework will be dependent on it. Many recent attacks are made by learning the key patterns. So to improve the security and to optimize the design, in this approach, the round keys which are required for encryption process will be generated using Random Number Generation unit (TRNG). As a further advancement, DNA encoding is used along with the TRNG design. A partial key of fewer bits is generated from the TRNG block and given to DNA encoder which will produce the complete 128-bits of key required. This approach will further strengthen the security and optimizes our design with reduced area and delay. The proposed design is implemented using Verilog coding, simulated by ModelSim 6.4 c and Synthesized using Xilinx 13.2 IDE with Virtex 5 FPGA as the target device. A comparison table is made for the standard approach, TRNG based AES approach and TRNG and DNA based AES.*

Key Words: **AES-128, TRNG, DNA encoding, Post processor, Pre-processor.**

1. INTRODUCTION

In the current digital era securing the data generated and transmitted over the web via cryptography is vital. Cryptography, normally called encryption is a process where the user data is encrypted into an unreadable format so that it can be protected from unwanted users. The intended recipients can access or read the data by decrypting it with proper calculations and keys. The National Institute of Standards and Technology called NIST section of the US government in the year 1977 needed an alternative for DES abbreviated as Data Encryption Standard which were then prone to attacks and also because of advances in processing power of the systems. Among many algorithms proposed and reviewed Joan Daemen and Vincent Rijmen, Belgium cryptographer's algorithm was chosen and named it as Rijndael algorithm. Later in the year 2000, it was formally adopted with the name Advanced Encryption Standard (AES) and was published as FIPS-197 under the federal standards [1].

AES operation uses symmetric key which implies that it will use the key that is same for both encryption and decryption process. There is flexibility in this algorithm to choose the size of input and key size among 128, 192 or 256-bit. In AES we keep the 128 bit as fixed size of input block while varying the size of key among 128, 192 or 256-bits. AES-128 name for the algorithm with 128 bit key, similarly AES-192 and AES-256 are the standard names used [1].

1.1 AES Algorithm

In the AES implementation, the 128-bits of input will be made into a group of 16 bytes block. The arrangement of the blocks will be in a matrix form consisting of four rows and columns. AES will do the calculations on these bytes of data instead on bits. For encrypting a block of data the number of rounds or iterations are not fixed as in DES, it relies on size of the key used. AES with 128-bit key uses 10 rounds, for key size of 192-bits 12 rounds, 14 rounds for AES with key of 256-bits [1]. Every iteration during encryption gets a new set of keys called round keys which is calculated using the initial given key and previous generated key as we go on. Here "rounds" imply that the AES algorithm will perform the mixing of input data re-encrypting it 10 to 14 times on the basis of key size. The process of decryption is simply an inverse of encryption process, all the steps which are done for encryption process is done in reverse order to decrypt the cipher data. The final round inversion is done first then the nine main rounds inversion and then the initial round inversion.

In the AES algorithm, the first round of operation is the Add round key where the input plain text block is XORed with the given cipher key. Then we have nine main rounds of calculations which have four stages in each round and the final round will have only three stages, this is same for decryption also, but it will just be the inverse operation of encryption. The 4 stages of operation are:

1. Substitute bytes
2. Shift rows
3. Mix Columns
4. Add Round Key

For the nth final round Mix column operation is not done [1].

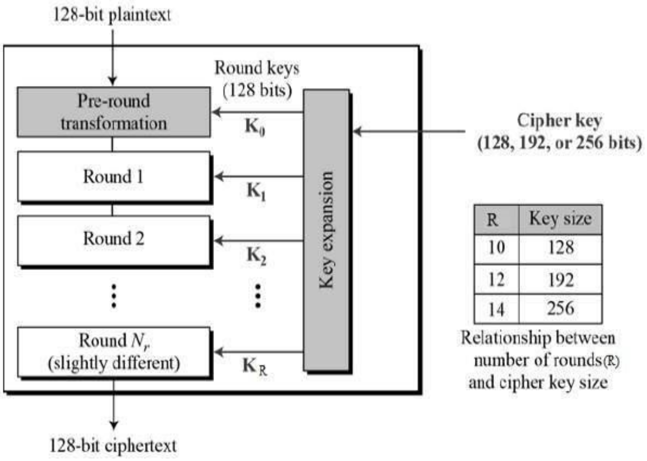


Fig -1: AES Encryption Process

2. TRULY RANDOM NUMBER GENERATOR

Random Number Generators are computational devices or physical device that will generate a series of numbers which will not have any dependencies or a visual patterns, such a series of numbers can be considered as random numbers. Such capacities might be portrayed as "True Random Number Generators" (TRNGs). Random numbers frame an essential piece of most security frameworks. Their most evident utilization is in the era of cryptographic keys for information encryption, the key which are generated here cannot be guessed or calculated even when we can get the previous keys or part of the key also - the more arbitrary the key, the more secure the framework. This is an ideal application for a fantastic TRNG. There are two stages which the RNG's include, a True Random Number generator that creates the entropy, and cryptographic post-handling, used to get a specific level of security even on account of an undetected disappointment of the basic TRNG.

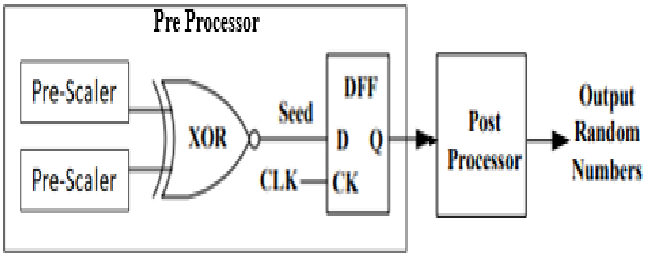


Fig -2: Random Number generator Block

Here a pre-scalers is made use of to implement the oscillator required to get clock pulses of different frequencies in the pre-processor stage. A 4/5 pre scaler is used to get two different clock pulses. A mode control bit MC controls the output depending on its value.

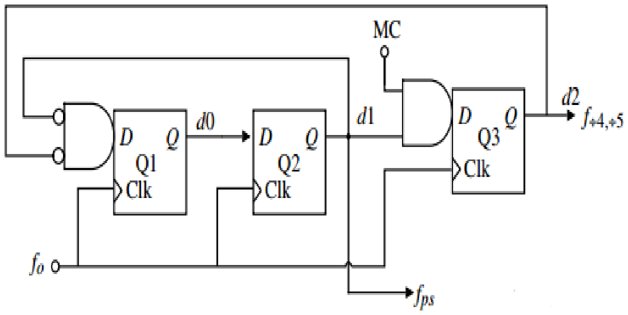


Fig -3: 4/5 Pre scaler circuit

In the preprocessor block the two outputs from the pre scalers are xored and inverted and given as a seed to the sampler which is a D-Flip flop. The output of this is given to a post processor as a clock which implements a LFSR circuit to increase the randomness of the output [3].

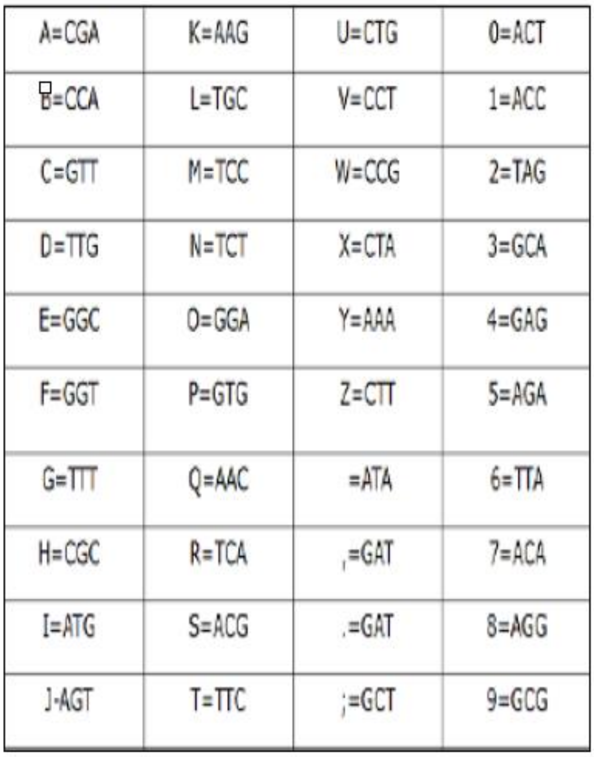
The TRNG includes random seed generator pre block and a post processor producing the final output. An important function of the post-digital processor is to give robustness of the statistical properties of the TRNG output sequence. The post digital processor is realized by 128-bit Linear Feedback Shift Register (LFSR) [3].

3. DNA ENCODING

DNA cryptography is one of the fast improving innovation which takes a shot at ideas of DNA processing. Another procedure for securing data was shown utilizing the cellular structure of DNA called DNA Computing. DNA can be utilized to store and transmit information. The idea of utilizing DNA computation in the fields of cryptography has been distinguished as a conceivable innovation that may present another desire for unbreakable algorithms.

DNA Strands are long polymers of a million number of connected nucleotides. These nucleotides comprise of one of four nitrogen bases, a five carbon sugar and a phosphate gathering. The nucleotides which make up these polymers are named after the nitrogen base that it comprises of; ACGT Adenine, Cytosine, Guanine, and Thymine [11].

Speed, less storage, minimal power requirements etc. are some of the advantages of DNA encoding. In DNA coding we can see that the input information that has to be encoded contains characters. The encoding unit takes this input data and generates a triplet code which will include a combo of three bases of DNA.



A=CGA	K=AAG	U=CTG	0=ACT
B=CCA	L=TGC	V=CCT	1=ACC
C=GTT	M=TCC	W=CCG	2=TAG
D=TTG	N=TCT	X=CTA	3=GCA
E=GGC	O=GGA	Y=AAA	4=GAG
F=GGT	P=GTG	Z=CTT	5=AGA
G=TTT	Q=AAC	=ATA	6=TTA
H=CGC	R=TCA	,=GAT	7=ACA
I=ATG	S=ACG	.=GAT	8=AGG
J-AGT	T=TTC	;=GCT	9=GCG

Fig -4: DNA Encoding Table

4. PROPOSED WORK

The design of a truly random number generator (TRNG) macro-cell to get the random keys of 128 bits, suitable to be integrated into AES Cryptographer, is done. The round keys are generated using this block instead of the Key Expansion process. The round keys obtained from the TRNG block will be stored in memory simultaneously performing the Encryption and decryption

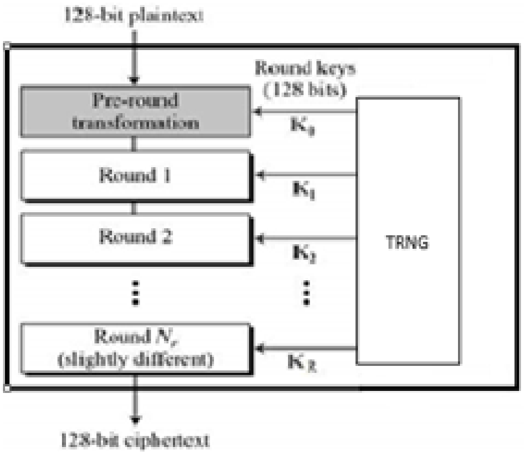


Fig -5: TRNG based AES Design

The complex process of the key expansion which include sub-byte, word rotation and Xor with R-Con is replaced with simple, efficient TRNG block. This proposed design will increase the security level as the key process will be fully random without any relation between the keys generated. The area and delay of the proposed system will be reduced.

Another approach proposed is to generate the round keys with the help of TRNG block and DNA encoding. In this approach to make sure that the keys obtained are fully random and impossible to guess we make use of the DNA encoding. Here we generate only partial bits from the TRNG block since the DNA encoding will produce triplet code for each input. The required 128 bits of key can be obtained by just giving 24 bits of input to DNA encoding. So the TRNG block can be made smaller for just producing 24 bits of output. This will further reduce the area and delay of the entire system.

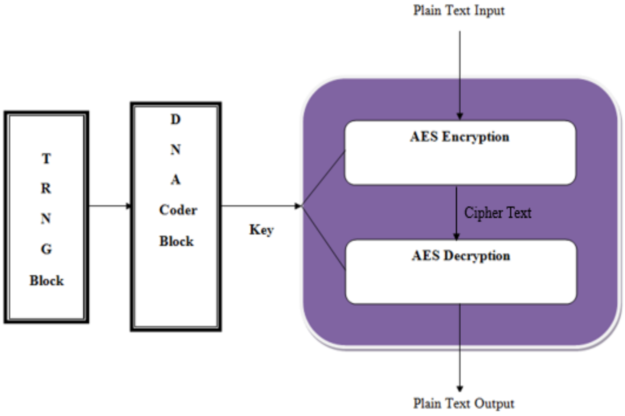
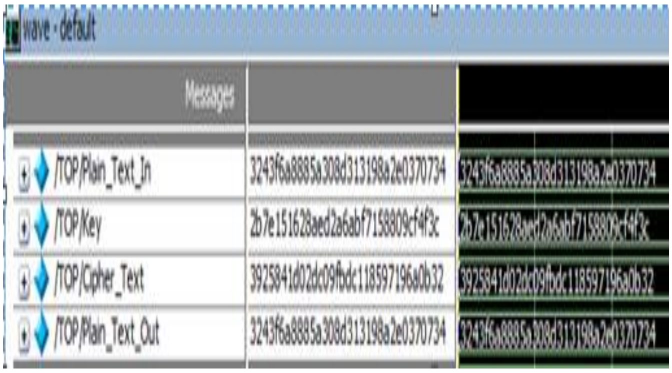


Fig -6: TRNG and DNA based AES Design

5. SIMULATION RESULTS

The system is designed in Verilog and simulated using ModelSim 6.4 c, various simulation outputs are shown below.



```
/TOP/Plain_Text_In    3243f6a8885a308d313198a2e0370734    3243f6a8885a308d313198a2e0370734
/TOP/Key              2b7e151628aed2a6abf7158809cf4f3c    2b7e151628aed2a6abf7158809cf4f3c
/TOP/Cipher_Text      3925841d02dc09fbdc118597196a0b32    3925841d02dc09fbdc118597196a0b32
/TOP/Plain_Text_Out   3243f6a8885a308d313198a2e0370734    3243f6a8885a308d313198a2e0370734
```

Fig -7: Normal AES Encryption and Decryption

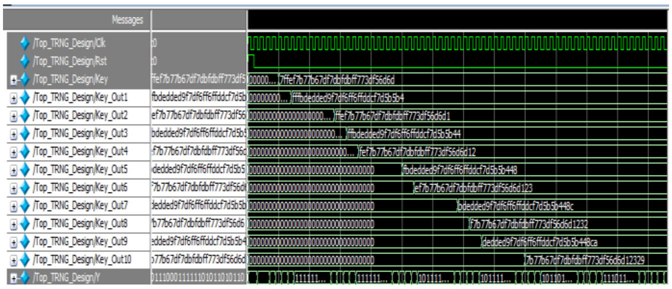

Messages
/Top_TRNG_Design/Clk 0
/Top_TRNG_Design/Rst 0
/Top_TRNG_Design/Key ffef7b77b67df7dbfdbff773df5 7ffef7b77b67df7dbfdbff773df56d6d
/Top_TRNG_Design/Key_Out1 fbdedded9f7df6ff6ffddcf7d5b fffbdedded9f7df6ff6ffddcf7d5b5b4
/Top_TRNG_Design/Key_Out2 ef7b77b67df7dbfdbff773df56 ffef7b77b67df7dbfdbff773df56d6d1
/Top_TRNG_Design/Key_Out3 bdedded9f7df6ff6ffddcf7d5b5 ffbdedded9f7df6ff6ffddcf7d5b5b44
/Top_TRNG_Design/Key_Out4 f7b77b67df7dbfdbff773df56c fef7b77b67df7dbfdbff773df56d6d12
/Top_TRNG_Design/Key_Out5 dedded9f7df6ff6ffddcf7d5b5 fbdedded9f7df6ff6ffddcf7d5b5b448
/Top_TRNG_Design/Key_Out6 7b77b67df7dbfdbff773df56d ef7b77b67df7dbfdbff773df56d6d123
/Top_TRNG_Design/Key_Out7 dedded9f7df6ff6ffddcf7d5b5b bdedded9f7df6ff6ffddcf7d5b5b448c
/Top_TRNG_Design/Key_Out8 b77b67df7dbfdbff773df56d6 f7b77b67df7dbfdbff773df56d6d1232
/Top_TRNG_Design/Key_Out9 edded9f7df6ff6ffddcf7d5b5b4 dedded9f7df6ff6ffddcf7d5b5b448ca
/Top_TRNG_Design/Key_Out10 77b67df7dbfdbff773df56d6d 7b77b67df7dbfdbff773df56d6d12329
/Top_TRNG_Design/Y 011100011111101011010110

Fig -8: TRNG Block output

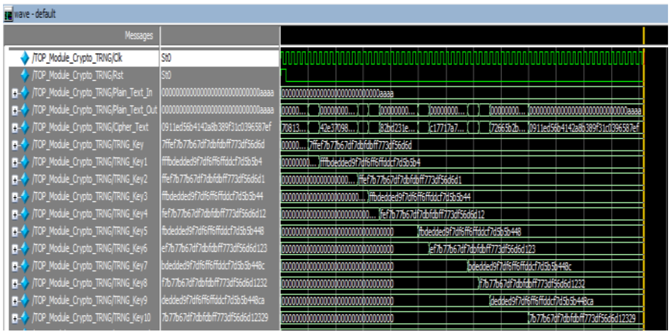

wave - default
Messages
/TOP_Module_Crypto_TRNG/Clk St0
/TOP_Module_Crypto_TRNG/Rst St0
/TOP_Module_Crypto_TRNG/Plain_Text_In 0000000000000000000000000000aaaa
/TOP_Module_Crypto_TRNG/Plain_Text_Out 0000000000000000000000000000aaaa
/TOP_Module_Crypto_TRNG/Cipher_Text 0911ed56b4142a8b389f31c0396587ef
/TOP_Module_Crypto_TRNG/TRNG_Key 7ffef7b77b67df7dbfdbff773df56d6d
/TOP_Module_Crypto_TRNG/TRNG_Key1 fffbdedded9f7df6ff6ffddcf7d5b5b4
/TOP_Module_Crypto_TRNG/TRNG_Key2 ffef7b77b67df7dbfdbff773df56d6d1
/TOP_Module_Crypto_TRNG/TRNG_Key3 ffbdedded9f7df6ff6ffddcf7d5b5b44
/TOP_Module_Crypto_TRNG/TRNG_Key4 fef7b77b67df7dbfdbff773df56d6d12
/TOP_Module_Crypto_TRNG/TRNG_Key5 fbdedded9f7df6ff6ffddcf7d5b5b448
/TOP_Module_Crypto_TRNG/TRNG_Key6 ef7b77b67df7dbfdbff773df56d6d123
/TOP_Module_Crypto_TRNG/TRNG_Key7 bdedded9f7df6ff6ffddcf7d5b5b448c
/TOP_Module_Crypto_TRNG/TRNG_Key8 f7b77b67df7dbfdbff773df56d6d1232
/TOP_Module_Crypto_TRNG/TRNG_Key9 dedded9f7df6ff6ffddcf7d5b5b448ca
/TOP_Module_Crypto_TRNG/TRNG_Key10 7b77b67df7dbfdbff773df56d6d12329

Fig -9: TRNG based AES

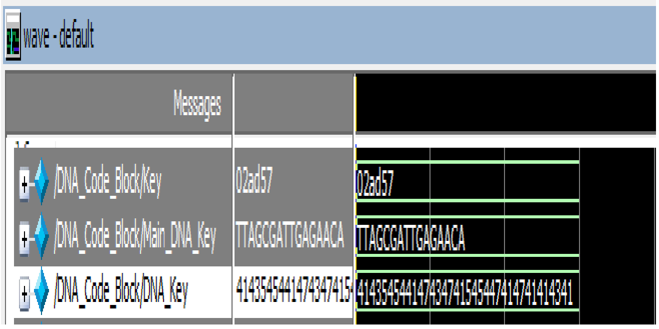

wave - default
Messages
/DNA_Code_Block/Key 02ad57
/DNA_Code_Block/Main_DNA_Key TTAGCGATTGAGAACA
/DNA_Code_Block/DNA_Key 414354544147434741545447414741414341

Fig -10: DNA Code Block

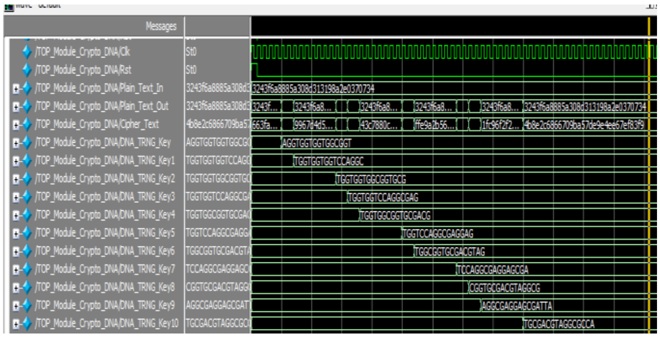

Messages
/TOP_Module_Crypto_DNA/Clk St0
/TOP_Module_Crypto_DNA/Rst St0
/TOP_Module_Crypto_DNA/Plain_Text_In 3243f6a8885a308d313198a2e0370734
/TOP_Module_Crypto_DNA/Plain_Text_Out 3243f6a8885a308d313198a2e0370734
/TOP_Module_Crypto_DNA/Cipher_Text 4b8e2c6866709ba57de9e4ee67ef83f9
/TOP_Module_Crypto_DNA/DNA_TRNG_Key AGGTGGTGGTGGCGGT
/TOP_Module_Crypto_DNA/DNA_TRNG_Key1 TGGTGGTGGTCCAGGC
/TOP_Module_Crypto_DNA/DNA_TRNG_Key2 TGGTGGTGGCGGTGCG
/TOP_Module_Crypto_DNA/DNA_TRNG_Key3 TGGTGGTCCAGGCGAG
/TOP_Module_Crypto_DNA/DNA_TRNG_Key4 TGGTGGCGGTGCGACG
/TOP_Module_Crypto_DNA/DNA_TRNG_Key5 TGGTCCAGGCGAGGAG
/TOP_Module_Crypto_DNA/DNA_TRNG_Key6 TGGCGGTGCGACGTAG
/TOP_Module_Crypto_DNA/DNA_TRNG_Key7 TCCAGGCGAGGAGCGA
/TOP_Module_Crypto_DNA/DNA_TRNG_Key8 CGGTGCGACGTAGGCG
/TOP_Module_Crypto_DNA/DNA_TRNG_Key9 AGGCGAGGAGCGATTA
/TOP_Module_Crypto_DNA/DNA_TRNG_Key10 TGCGACGTAGGCGCCA

Fig -11: TRNG and DNA based AES

6. SYNTHESIS RESULTS

The design is synthesized by Xilinx tool 13.2 IDE with Virtex 5 XC5vlx110t-1ff1136 FPGA as the target device.

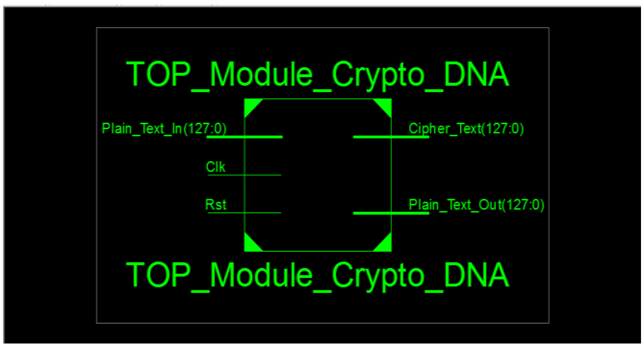


Fig -12: RTL Schematic of Top Module Design

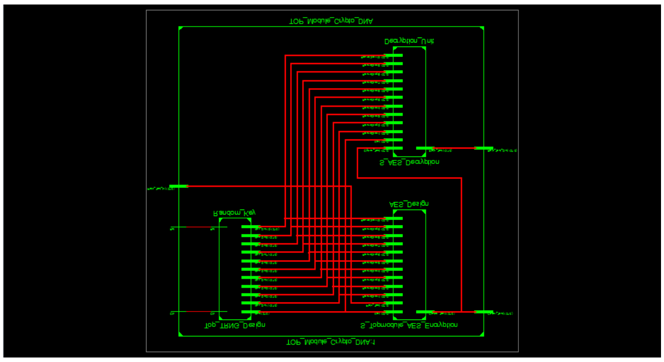
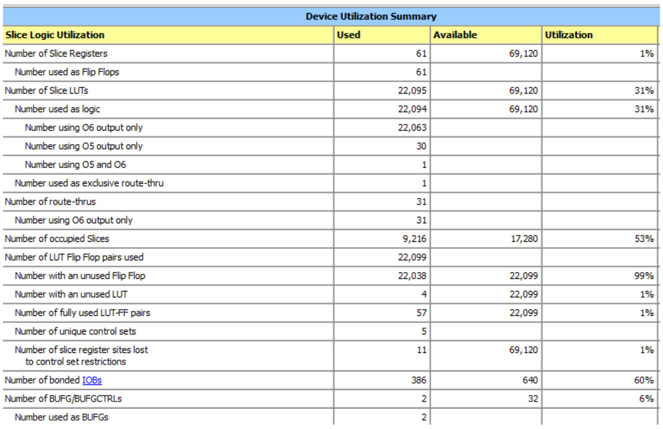
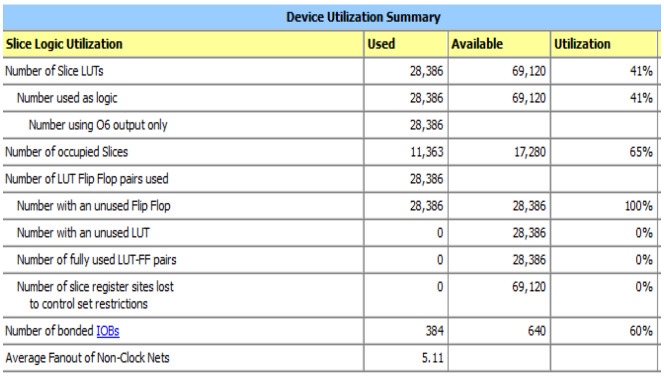


Fig -13: RTL Schematic of Inner Modules



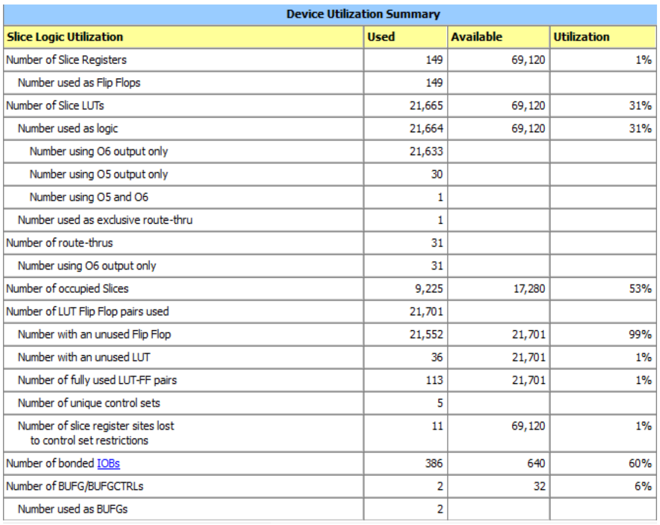
Device Utilization Summary			
Slice Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	61	69,120	1%
Number used as Flip Flops	61		
Number of Slice LUTs	22,095	69,120	31%
Number used as logic	22,094	69,120	31%
Number using O6 output only	22,063		
Number using O5 output only	30		
Number using O5 and O6	1		
Number used as exclusive route-thru	1		
Number of route-thrus	31		
Number using O6 output only	31		
Number of occupied Slices	9,216	17,280	53%
Number of LUT Flip Flop pairs used	22,099		
Number with an unused Flip Flop	22,038	22,099	99%
Number with an unused LUT	4	22,099	1%
Number of fully used LUT-FF pairs	57	22,099	1%
Number of unique control sets	5		
Number of slice register sites lost to control set restrictions	11	69,120	1%
Number of bonded IOBs	386	640	60%
Number of BUFG/BUFGCTRLs	2	32	6%
Number used as BUFGs	2		

Fig -14: Device utilization summary of TRG based AES



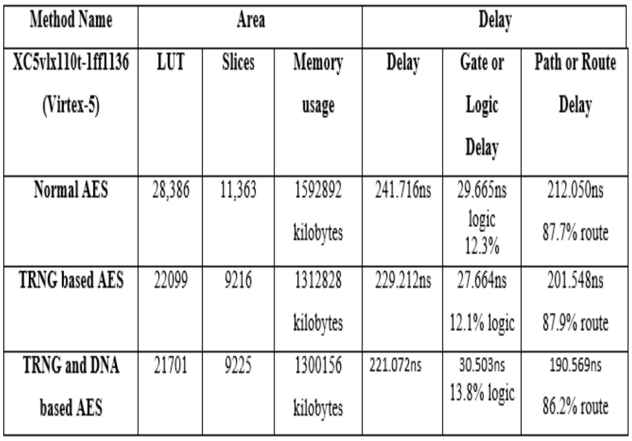
Device Utilization Summary			
Slice Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice LUTs	28,386	69,120	41%
Number used as logic	28,386	69,120	41%
Number using O6 output only	28,386		
Number of occupied Slices	11,363	17,280	65%
Number of LUT Flip Flop pairs used	28,386		
Number with an unused Flip Flop	28,386	28,386	100%
Number with an unused LUT	0	28,386	0%
Number of fully used LUT-FF pairs	0	28,386	0%
Number of slice register sites lost to control set restrictions	0	69,120	0%
Number of bonded IOBs	384	640	60%
Average Fanout of Non-Clock Nets	5.11		

Fig -15: Device utilization summary of Normal AES



Device Utilization Summary			
Slice Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	149	69,120	1%
Number used as Flip Flops	149		
Number of Slice LUTs	21,665	69,120	31%
Number used as logic	21,664	69,120	31%
Number using O6 output only	21,633		
Number using O5 output only	30		
Number using O5 and O6	1		
Number used as exclusive route-thru	1		
Number of route-thrus	31		
Number using O6 output only	31		
Number of occupied Slices	9,225	17,280	53%
Number of LUT Flip Flop pairs used	21,701		
Number with an unused Flip Flop	21,552	21,701	99%
Number with an unused LUT	36	21,701	1%
Number of fully used LUT-FF pairs	113	21,701	1%
Number of unique control sets	5		
Number of slice register sites lost to control set restrictions	11	69,120	1%
Number of bonded IOBs	386	640	60%
Number of BUFG/BUFGCTRLs	2	32	6%
Number used as BUFGs	2		

Fig -16: Device utilization summary of TRNG and DNA based AES



Method Name	Area			Delay		
XC5vlx110t-1ff1136 (Virtex-5)	LUT	Slices	Memory usage	Delay	Gate or Logic Delay	Path or Route Delay
Normal AES	28,386	11,363	1592892 kilobytes	241.716ns	29.665ns logic 12.3%	212.050ns 87.7% route
TRNG based AES	22099	9216	1312828 kilobytes	229.212ns	27.664ns 12.1% logic	201.548ns 87.9% route
TRNG and DNA based AES	21701	9225	1300156 kilobytes	221.072ns	30.503ns 13.8% logic	190.569ns 86.2% route

Fig -17: Comparision Results

6. CONCLUSION

The design of AES with TRNG which is used for generation of the round keys required for encryption and decryption process, also DNA based TRNG module is implemented to increase the security level further is implemented. The Implementation is based on mathematical properties of Rijndael algorithm where the key expansion part relies on DNA Coder and TRNG. Encryption Design using Shit rows, Mixed Column, Add Round Key is done, and Design of Decryption Part is also done. Genetic Algorithm based Encoding for Key Generation is used for Encryption and Decryption Process. The new design permits the construction of efficient area and speed characteristics, while still keeping a very high protection level. We conducted relevant AES Implementation with DNA TRNG for Key Generation Method. With this novel approach of generating the random keys which is purely random and cannot be guessed and also the key entry process will no longer be a manual process. All the set of 128-bit keys required will be generated randomly. The level of security against various attacks will be increased using this method. Also, from the comparison table we can see that the area and the delay when compared to the conventional AES implementation has been reduced. The design has an enormous scope of improvement; the TRNG block can be implemented using various other methods. A true source of randomness can be added also there are a variety of true sources of randomness are available and which may help in further optimizing the design.

REFERENCES

[1] Announcing the Advacned Encryption Standard (AES), Federal Information processing Standards Publication 197,http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf.

[2] Practical Implementation of Rijndael S-Box Using Combinational Logic, Edwin NC Mui Custom R & D EngineerTexcoEnterprisePtd.Ltd.http://www.xess.com/static/media/projects/Rijndael_SBox.pdf.

[3] Liu Dongsheng, Liu Zilong, Li Lun*, Zou Xuecheng “A Low-Cost Low-Power Ring Oscillator-Based Truly Random Number Generator for Encryption on Smart Cards”, IEEE Transactions on Circuits and Systems II: Express Briefs (Volume: 63, Issue: 6, June 2016)

[4] Apostol Vassilev,Timothy A. Hall, “The Importance of Entropy to Information Security”, IEEE COMPSAC 2014.

[5] Differential powerAnalysis: A serious threat for FPGA security, M. Masoumi, Int. J. Internet Technol. Secured Trans., vol. 4, no. 1, pp. 12–25, 2012.

[6] Naveen Jarold K, “Hardware Implementation of DNA Based Cryptography”,Proceedings of 2013 IEEE Conference on Information and Communication Technologies (ICT 2013).

[7] https://en.wikipedia.org/wiki/Affine_transformaton

[8] https://en.wikipedia.org/wiki/Cryptography

[9] https://www.designreuse.com/articles/27050/true-randomness-in-cryptography.html

[10] http://securityaffairs.co/wordpress/33879/security/dna-cryptography.html

[11] http://resources.infosecinstitute.com/dna-cryptography-and-information-security/#gref