FPGA Implementation of Modified AES Algorithm for Improved Timing

[1] Qazi Sama, [2] Er. Sandeep Sangwan

[1]Mtech Scholar, [2]Assstant Professor
ECE Department, SDDIET, Barwala, Haryana, India

***Abstract*—This Paper presents Pipelined and LUT based crypto multiplication implementation of high speed AES algorithm using Verilog. The proposed architecture is based on optimizing timing in terms of adding inner and outer pipeline registers for each rounds and Key Expansions. Further by optimizing the Crypto Multiplication for Mix columns via LUT based approach aid in further optimization in terms of timing, LUT and Pipelined based implementation techniques are optimal for FPGA based implementations. ROM table and pipelining are the two techniques used to implement AES. With the use of fully pipelined architecture and Distributed/Split LUT-Pipelined techniques, the throughput and speed of the encryption is increased tremendously. Xilinx ISE 14.7 ISE is used for synthesis and simulation of this proposed architecture. Xilinx Vivado can also be used to obtain results for ultra-scale devices, Implementation results are obtained for a Spartan6 Family of FPGA.**

***Index Terms*—AES, Sub bytes, Mix columns, LUT, Verilog HDL, FPGA, Speed, Pipelined, crypto multiplication**

I. INTRODUCTION

In the recent years, we have witnessed the increasing deployment of applications with a crucial need for security functions such as confidentiality, authentication, non-repudiation and time-stamping. These include for example e-commerce, secure e-mail, e-banking and other security functions. A cryptographic algorithm works with a key — a word, number, or phrase — to encrypt the plain text. The same plaintext encrypts to different cipher text using different keys. The security of encrypted data mainly depends on two things: the strength of the cryptographic algorithm and the secrecy of the key. A cryptographic algorithm is all possible keys and all the protocols that make it work comprise a cryptosystem. In conventional cryptography, also called secret-key or symmetric-key encryption, single key is used both for encryption and decryption. In asymmetric cryptography, the encryption and decryption keys are different on both the sides. We have implemented RC6 Algorithm.

AES is the Advanced Encryption Standard, an US government standard for encrypting and decrypting text. This standard is delineated in Federal information science normal (FIPS) 1971.NIST wanted to “make alternatives that supply a best level of security”, compared to the information encoding normal (DES), that grew susceptible to brute-force attacks as a result of its 56-bit effective key length [1][2]. An interchangeable block cipher that supported multiple key lengths was needed by AES candidates. The National Institute of Standards and Technology (NIST) is [3] published request for comments for the “Development of a Federal Information Processing Standard for AES, On January 2, 1997.

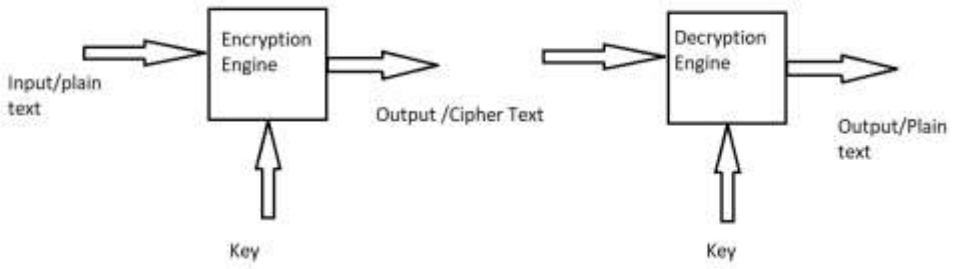


Fig 1: Cryptography system

II. AES ALGORITHM

AES is a non-Feistel cipher that encrypts and decrypts a data block of 128 bits. It uses 10, 12, or 14 rounds in algorithm. The key size, which can be 128, 192, or 256 bits depends on the number of rounds. AES uses four types of transformations to provide security.

III. Proposed AES Algorithm Implementation Technique

The pipelined architecture implementation is a modification of the iterative looping architecture. Registers are included after every round this is the external pipelining and each module like mix columns, key expansion, and Sbox also are pipelined internally to take leverage and reduce the overall critical path. Further Sbox and crypto multiplication is implemented via multiple LUT based implementation and a pipelined register in between. These registers help us in achieving the pipelining of the AES. Basically pipelining means to process the data that is given as input in a continuous manner without having to wait for the current

Process to get over. This pipelining concept is seen in many processors.

In this architecture, the registers are used to store the current output of the round that is being executed. Now instead of passing the output of each round to the next round directly we use a register which would act as a bypass or an internal register. Since the current rounds' value is stored in the register the next input to the current round can be given as soon as the current output is obtained. And the input to the next round is given from the register thus avoiding a direct contact between the two rounds. This is not possible in the iterative looping architecture because the next input can be given only when the whole round-based processing is over since the same hardware is used over and again in the process of obtaining the cipher text. Thus, the pipelined architecture increases the speed of execution for obtaining the cipher text but at a cost of area and clock latency.

1) Pipelining of Key Schedule Algorithm

Apart from Encryption and Decryption Module, another main component is Key Expansion Schedule. The security factor of the AES Encryption / Decryption Standard mainly depends on this part. For better security, in AES Algorithm first round user key is XORed with the original Plain / Cipher Text. And next round onwards Expanded Key from Expanded Key Schedule is XORed with data. The expansion algorithm of the AES is fixed. To speed up the process of Key Generation, it is preferable to opt for pipeline architecture. Key Expander comprises of EX-OR, Pipelined Data Registers. Since there are 44 words used in the key expansion process 44 data registers will be used, four in each stage of the pipeline. Registers are included between each round and thereby creating a sort of buffer between each round to provide the input without having to wait for the whole process to get over. So since the inputs are given at a faster rate and outputs are also obtained at a faster rate. So without pipelining the second output is obtained at the end of 22 cycles but with the help of pipelining the output is obtained at the end of 12 cycles thereby speeding up the process of obtaining the output.

2) Latency and Throughput

Apart from Encryption and Decryption Module, another main component is Key Expansion Schedule. The security factor of the AES Encryption / Decryption Standard mainly depends on this part. For better security, in AES Algorithm first round user key is XORed with the original Plain / Cipher Text. And next round onwards Expanded Key from Expanded Key Schedule is XORed with data. The expansion algorithm of the AES is fixed. To speed up the process of Key Generation, it is preferable to opt for pipeline architecture. Key Expander comprises of EX-OR, Pipelined Data Registers. Since there are 44 words used in the key expansion process 44 data registers will be used, four in each stage of the pipeline. Registers are included between each round and thereby creating a sort of buffer between each round to provide the input without having to wait for the whole process to get over. So, since the inputs are given at a faster rate and outputs are also obtained at a faster rate. So without pipelining the second output is obtained at the end of 22 cycles but with the help of pipelining the output is obtained at the end of 12 cycles thereby speeding up the process of obtaining the output.

3) Crypto Multiplication and Key Generation

Crypto Multiplication is part of Mix columns module of AES. As per Multiplication, in Cryptography there are specific rules for it. There are 3 Crypto multiplication i.e. with 1, 2 and 3 can be summed up with below implementations. These Operations can be internally pipelined to speed up the overall speed of the design. Overall latency will increase depending on the pipelined stages.

4) Distributed LUT – Pipelined Technique

As per previous architectures Sbox is implemented as a single LUT based implementation, with depth of 256 8 bit words', input to it is 8 bit data which acts as the address to LUT, output of LUT is the substituted value of input. S-box module is a crucial block which is used in 2 modules i.e. sub-bytes and Key expansion.

IV. IMPLEMENTATION BLOCKS

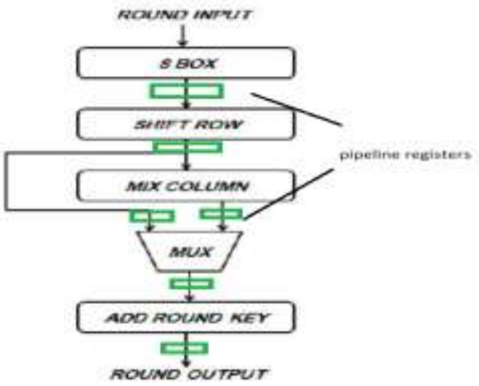


Fig2: Pipelining registers between Rounds

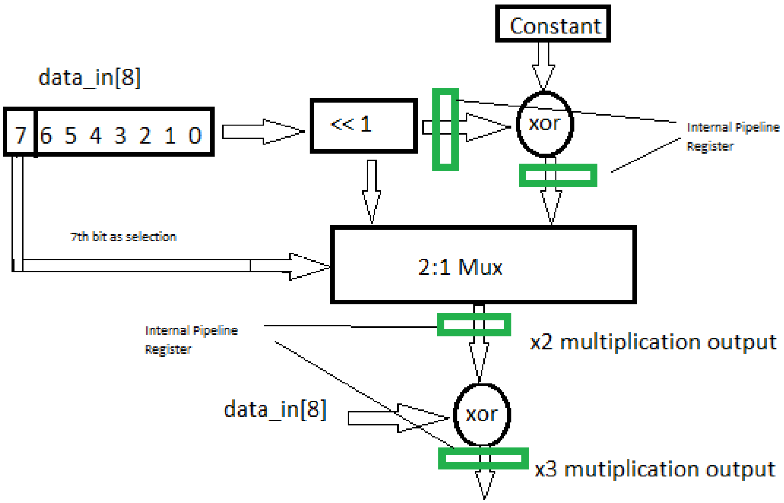


Fig3: Pipelined Crypto Engine

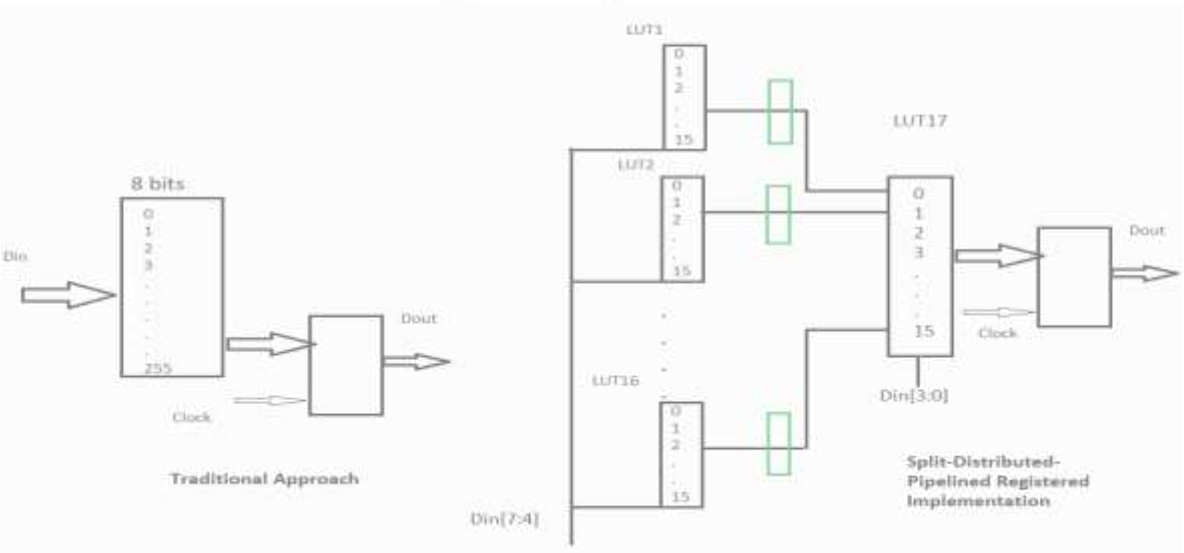


Fig4: Distributed LUT-Pipelined Block

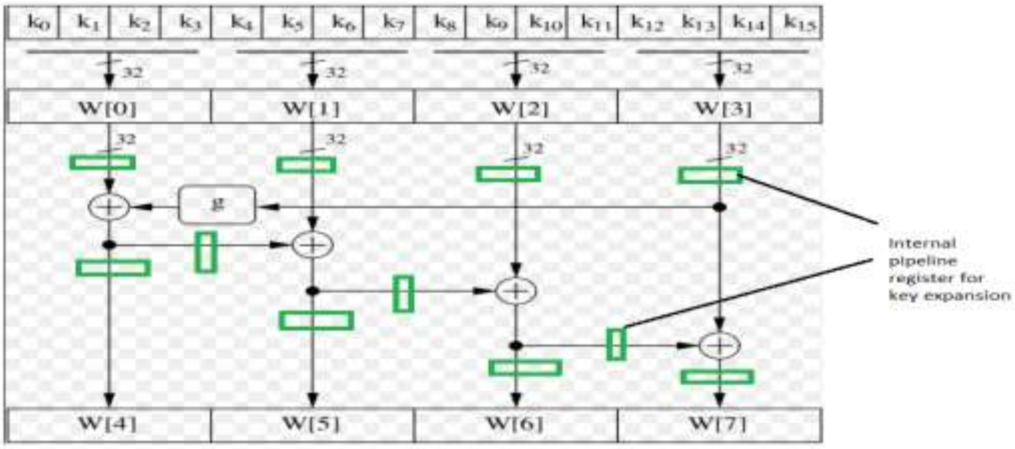
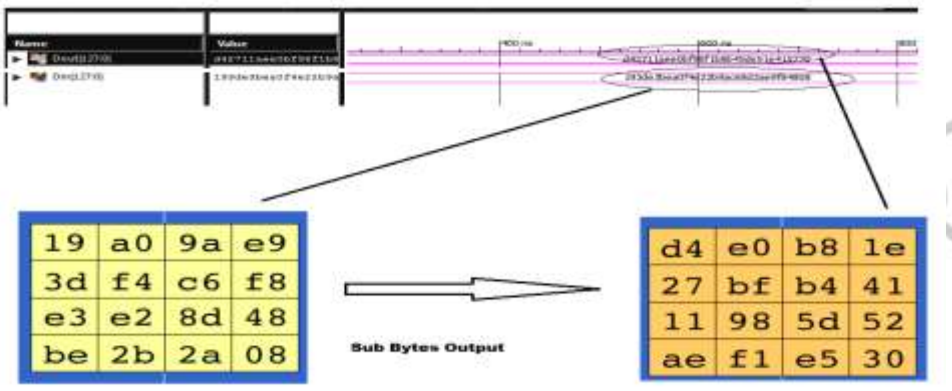


Fig5: Pipelined Key expansion

V. Simulation and Synthesis Results

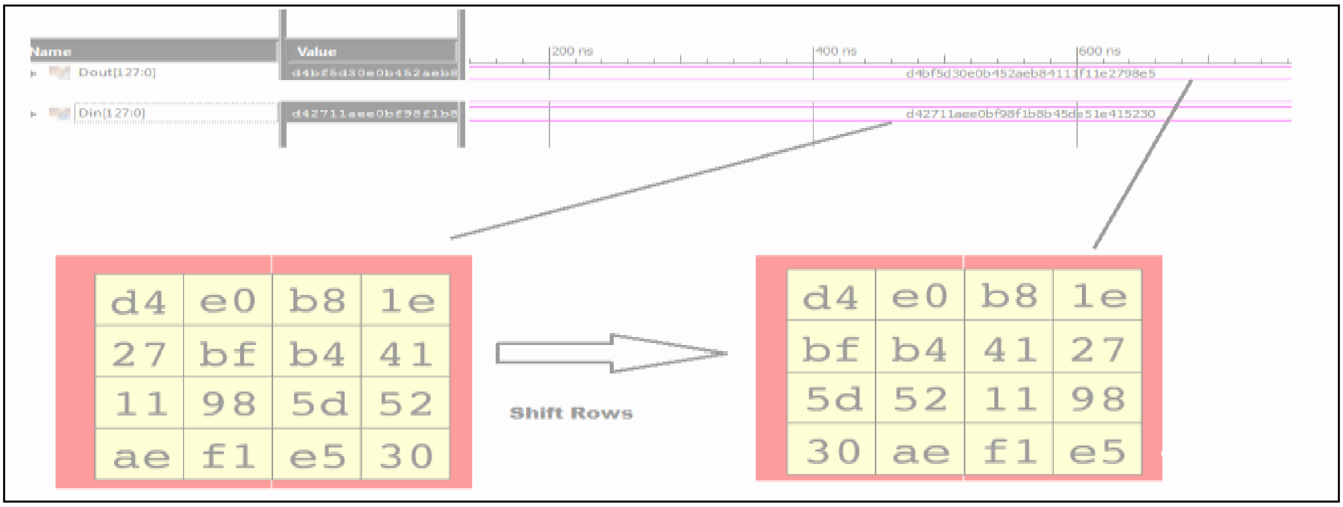
The simulation studies involve thorough analysis of the vectors for individual models as well as system level i.e top level block, below are the simulation results for individual model as well as top module.



19	a0	9a	e9
3d	f4	c6	f8
e3	e2	8d	48
be	2b	2a	08

d4	e0	b8	1e
27	bf	b4	41
11	98	5d	52
ae	f1	e5	30

Fig6: Sub bytes simulations



```
Dout[127:0]   d4bf5d30e0b452aeb8   d4bf5d30e0b452aeb84111f11e2798e5
Din[127:0]    d42711aee0bf98f1b8   d42711aee0bf98f1b8b45de51e415230
```

d4	e0	b8	1e
27	bf	b4	41
11	98	5d	52
ae	f1	e5	30

d4	e0	b8	1e
bf	b4	41	27
5d	52	11	98
30	ae	f1	e5

Fig7: Shift rows

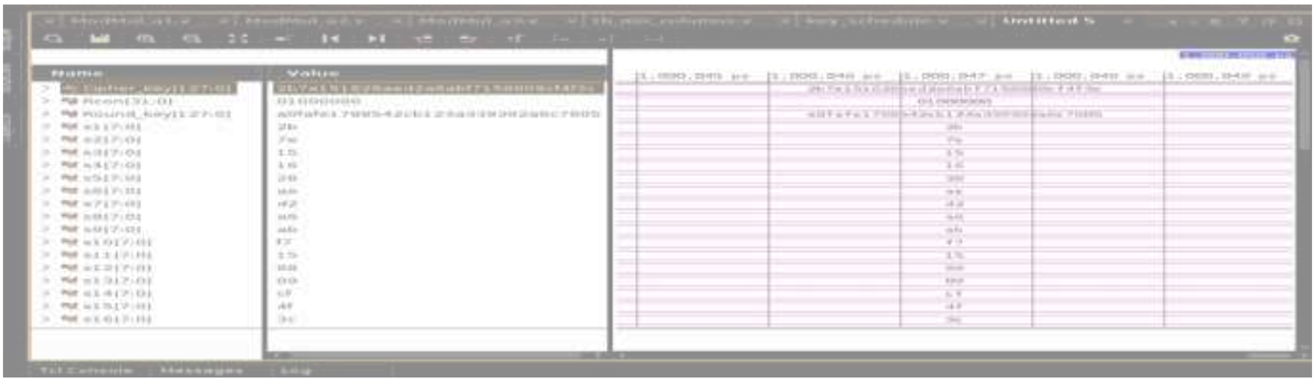


Fig8: Key expansion

Fig9: Top level encryption and decryption

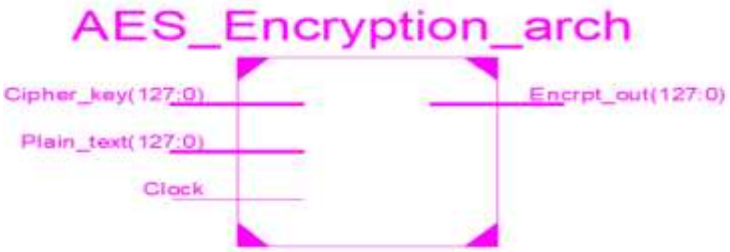


Fig 10: AES Top

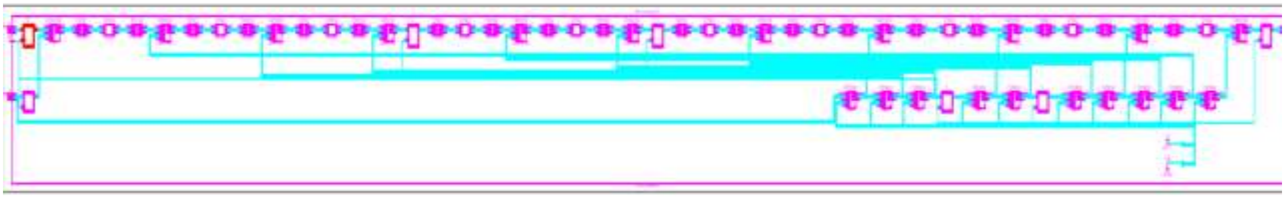


Fig 11: AES Internal

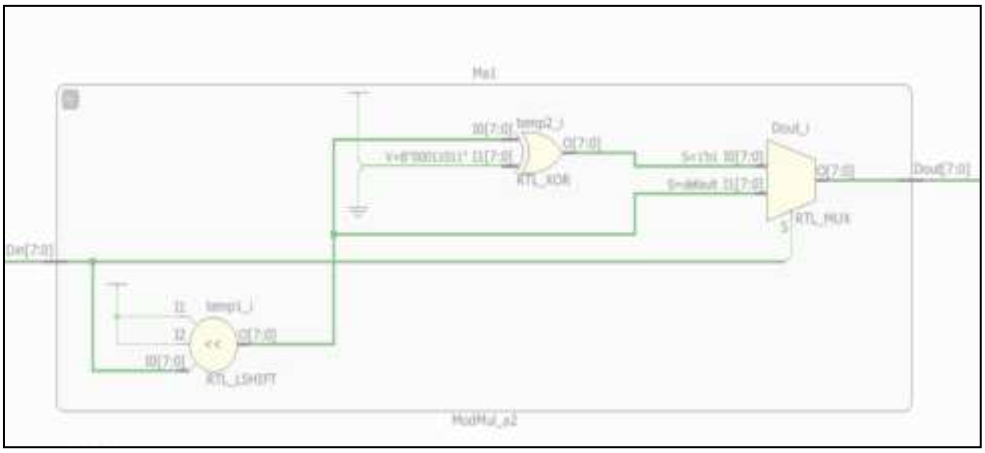


Fig 12: Mod Mul2

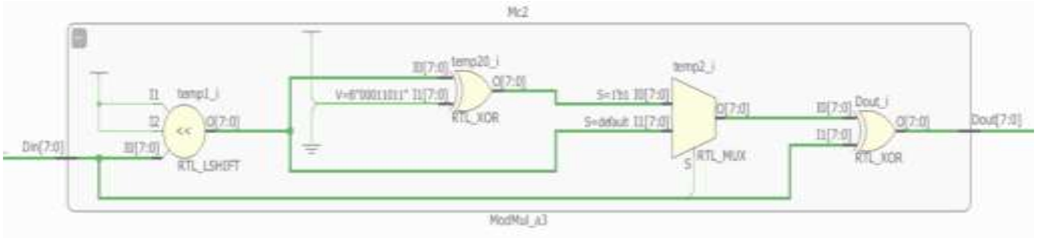


Fig13: Mod Mul3

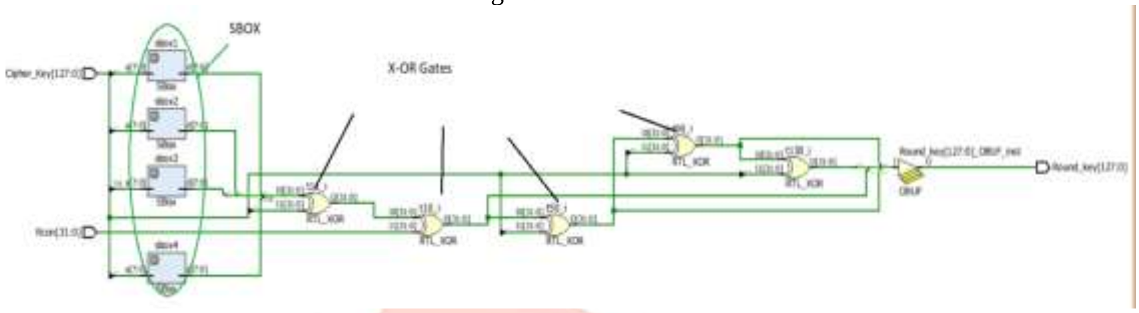
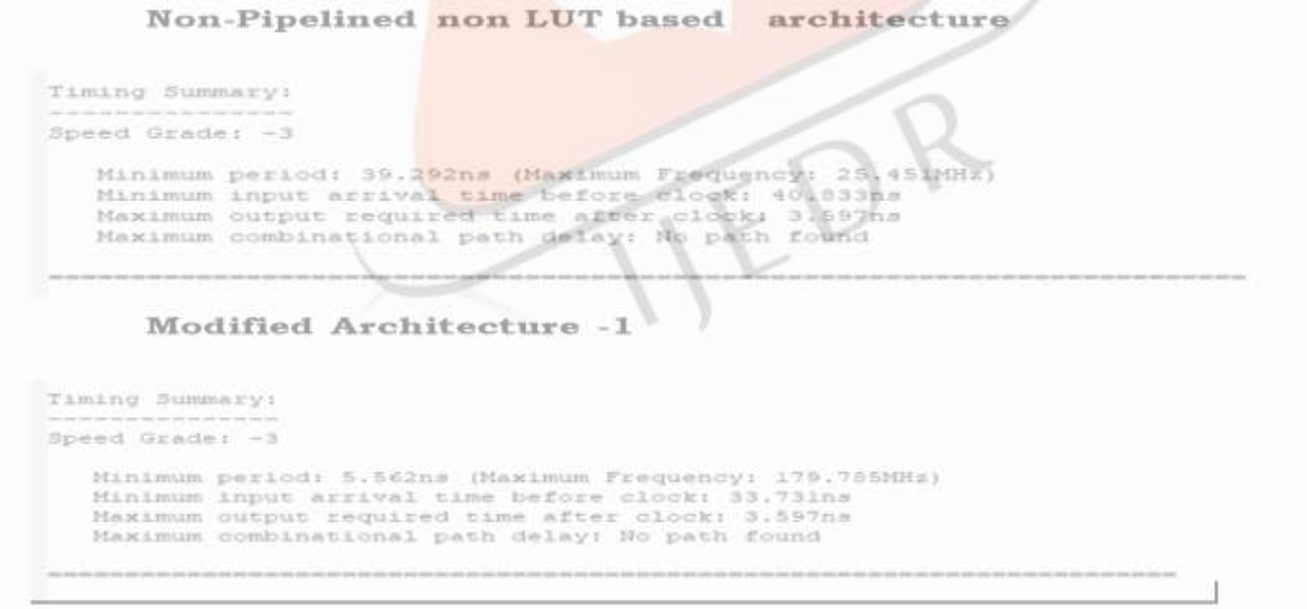


Fig14: Key Expansion

Comparison Results

Below are the timing results for Non-Pipelined, Non LUT based AES architecture and one with the modified one which has pipeline staged and LUT based. The FPGA Part selected to get implementation results is Family:Spartan6
Part No: xc6slx25t-3csg324

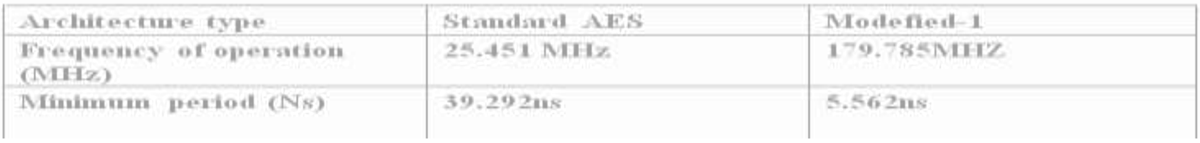


Non-Pipelined non LUT based architecture

```
Timing Summary:
---------------
Speed Grade: -3

   Minimum period: 39.292ns (Maximum Frequency: 25.451MHz)
   Minimum input arrival time before clock: 40.833ns
   Maximum output required time after clock: 3.597ns
   Maximum combinational path delay: No path found

=========================================================================
```

Modified Architecture -1

```
Timing Summary:
---------------
Speed Grade: -3

   Minimum period: 5.562ns (Maximum Frequency: 179.785MHz)
   Minimum input arrival time before clock: 33.731ns
   Maximum output required time after clock: 3.597ns
   Maximum combinational path delay: No path found

=========================================================================
```



Architecture type	Standard AES	Modefied-1
Frequency of operation (MHz)	25.451 MHz	179.785MHZ
Minimum period (Ns)	39.292ns	5.562ns

Fig15: Speed Comparisons

VI. Conclusion

The implementation results showed that the proposed algorithm performs better than the base algorithm with the total critical path getting further optimized therefore increasing the speed of operation. It can be clearly seen from the above synthesis results that in the fully pipelined AES encryption architecture the throughput are many folds greater than the conventional AES architecture and the single stage pipelining architecture (Fig15). The techniques used are Internal and outer Pipelining of modules and Distributed LUT based concept, the objective of the above techniques is to reduce the critical path delay and increase the overall speed of operation of design, the disadvantage is the increases in area and output latency. Area is not of a much problem as modern days FPGA's has huge amount of resources. Output latency needs to be taken care with extra logic whenever AES is integrated with other system modules.

References

[1] Daemen J., and Rijmen V, "The Design of Rijndael: AES-the Advanced Encryption Standard", Springer-Verlag, 2002 FIPS 197, “Advanced Encryption Standard (AES)”, November 26,2001

[2] Daemen J., and Rijmen V, "The Design of Rijndael: AES-the Advanced Encryption Standard", Springer-Verlag, 2002

[3] FIPS 197, “Advanced Encryption Standard (AES)”, November 26,2001

[4] J. Nechvatal, et. al., Report on the Development of the AdvancedEncryption Standard (AES), National Institute of Standards and Technology, October 2, 2000,at

[5] [4]AES page available via http://www.nist.gov/CryptoToolkit.

[6] J. Daemen and V. Rijmen, AES Proposal: Rijndael, AES Algorithm Submission, September 3, 1999, available at [1].