ISSN 2347 - 3983
Volume 9. No. 4, April 2021
International Journal of Emerging Trends in Engineering Research
Available Online at http://www.warse.org/IJETER/static/pdf/file/ijeter02942021.pdf
https://doi.org/10.30534/ijeter/2021/02942021

Advanced FPGA Implementation of AES Algorithm

Nitesh Kumar Dixit[1]
[1]Electrical Engineering Department, Bhartiya Institute of Engineering & Technology, Sikar, India
Nitesh20.dixit@gmail.com

ABSTRACT

Along with the enhance in computation as well as information safe-keeping in cloud servers, the requirement for a devoted computer hardware accelerator with regard to encryption is arising to be able to decrease the processor work. Highly efficient Advanced Encryption Standard (AES) 128-bit implementation, that could be utilized as an accelerator. In this research paper resource optimization and higher throughput were obtained. Memory segmentation is actually carried out to be able to assign several ports for simultaneous information accessibility. When algorithm is in proceed and examined time delay and initiation time period of various procedures, with regard to every crucial route delay, a fresh multistage solitary initiation time period sub-pipelined structure is suggested for making the initiation time period to a single for smallest route latency. Consequently, almost all operations in AES could be started within a single clock cycle and also can easily accept input in each and every clock cycle. The suggested approach while examined on latest Field Programmable Gate Array (FPGA) XC7VX690T unit that offers a throughput of 104.06 Gbps at a highest frequency of 813MHz and also 1.23-ns route delay. The useful resource utilization is reduced whenever compared along with other alternatives. The suggested method offers 30.74Mbps efficiency on device, which usually was 27.13% much more compared to the best efficiency documented in an earlier research study.

Key words: Advanced Encryption Standard, Cryptography, Field Programmable Gate Array, Resource Optimization, Throughput.

1. INTRODUCTION

The algorithm formulated by Joan and Vincent in 2001. It's safe and secure in characteristics, versatility, and simplicity of implementation [1,2] and it can be carried out in software or hardware. The level that on which the software program implementation can be improved is restricted by the underlying equipment. Explicit hardware modifications should not be carried out whenever the algorithm is applied to software. These restrictions can be managed utilizing loop unrolling, pipelining, parallelism, and so on. And this hardware configuration can be easily done in Field programmable gate arrays (FPGAs) [1,3,4]. As every area of living begins transferring on the internet, generally there arises the requirement for storage and protection of massive amount of consumer information. Encryption and decryption are generally a couple of major operations which requires large amount of processing moment. Initially, almost all cryptographic procedures were performed through the processor itself, that restricted the overall functionality of the program and also caused an improve in the power generation.

Devoted equipment emerged into existence for performing encryption/decryption on servers. Effective implementation of AES by managing resources and overall performance is one particular problem that must be taken attention of. The requirement for a higher speed encryption in the particular form of accelerator in equipment came into image together with the introduction of cloud computing. Encryption has been utilized in cloud, whilst the information tends to be in utilize and at rest. Full consumer virtual machine (VM) encryption and decryption, traffic into and out of VMs whilst operating, encryption as a service in cloud, and so on, needs higher throughput with a lesser amount of resource utilization [4-6]. Numerous cloud companies like Amazon, Google Cloud, CloudLink, and CloudSigma use encryption, information targeted traffic encryption, important safety, and so on [8-13]. The encryption providers on cloud are mostly dependent on time frame and space. The advancement of accelerator on hardware FPGA can create the encryption a lot more quickly, consume fewer resource, minimize the processor work, as well as minimize the power to a significant level. As FPGAs need significantly less frequency compared to processor chip, heat production will probably be really less. Whenever the primary processor requirements and more rapid assistance for encryption, its lookups the bitstream storage space for the specific equipment design. When the equipment design is located, the related bitstream is packed directly into attached FPGA. The application is actually then operating over the equipment and gets the outcome back again to primary processor. The bitstream or equipment layout can be acquired from the equipment developer or through third-party IP suppliers. In the suggested technique, we personalize and improve the resources in FPGA, enhancing the pipelining performance, and therefore, greater throughput is accomplished with fewer number of resources. Memory space dividing approach is utilized to permit reading

through and composing of input and output information in parallel, that enhances the pipelining. Based on the examined parameters, a fresh multistage pipelining which tends to make the initiation period of all procedures and the whole system to be a single clock cycle is suggested of 813MHz Clock Frequency. Merging of a couple of functions is performed to prevent unwanted delay and unnecessary resource utilization. Mapping as well as placement of resources is carried out optimally, for this purpose AES-ECB mode and AES-Counter mode both are utilized.

2. LITERATURE REVIEW

Farashahi et al. [14] suggested a 2-slow retiming approach that expands the c-slow retiming approach for the throughput enhancement of AES algorithm. The c-slow retiming approach enhances the pipelining by splitting the data pathways and transferring the registers at particular locations to enhance the structures. The 2-slow retiming approach supersedes each and every register in c-slow retime approach with a couple of registers. Information forwarding is utilized to eliminate dependency mistakes in c-slow time approach. This approach offered throughput of 86 Gbps at 671.524MHz. An additional program is created along with fast AES design and style utilizing LookUp Tables (LUTs), as well as it provides additional safety for AES core [15]. A function creator is employed to load the LUT material on the base of the several variables of AES, that offers additional safety [15].

In the research by Liu et al. [16], in the beginning the experts examined the logic detail of combinational circuits employed in each and every of the AES procedures. To decrease the logic depth equipment layout, a couple of phase pipelining method and deep pipelining techniques were employed. It might accomplish a throughput of 75.9 Gbps. A fresh Key Expansion scheme which improved the complexness up to 2 (N −1) had been also suggested. An additional research [17] showed various techniques for applying s-box of AES in numerous suggested styles. The creators utilized loop unrolling for crucial route customization and completely pipelined and sub-pipelined methods, that permitted the improve in clock frequency and decrease in crucial route. Optimum sub-pipeline register quantity and locations were discovered by considering all opportunities. Composite field and combinational logic strategy for s-box execution are compared within the foundation of area and throughput. To decrease the employed area, Lee et al. [18] employed a sequence of continuous binary matrix multiplication rather of Galois field (GF) (28) calculation. Additionally, a four-step pipelined execution is also offered. Together, a couple of techniques propose for decrease in area along with for greater throughput. Benaissa [19] introduced a couple of patterns for AES feedback mode assistance. Initially layout allocated LUTs among pipeline slashes to accomplish optimum throughput. Second style concentrated on Key Expansion utilizing which key may be transformed each and every clock cycle by appropriate pipelining. The strategy offered in Hammad et al. [20] divides and rearranges the functioning in AES to attain the best possible area and throughput. The Mix Column procedure is divided and rearranged along with Add Round Key procedure. The throughput achieved of 39,053Mbps at clock frequency 305.1MHz [20,21].

A combined-block strategy employing Key Expansion design for improving the throughput and for minimizing the power usage was introduced by Kalaiselvi [22]. In numerous additional scientific studies [23-27], various degrees of pipelined architectures were suggested. Granado et al. [28] created a high-speed AES formula execution for Wi-FI Protected Access 2 (WPA2) systems. It utilized powerful and partial reconfiguration together with parallelism. The particular program was demonstrated to provide a 24.922 Gbps of throughput. Glesner [29] introduced AES setup for tiny gadgets that work at 50MHz in Offset Codebook (OCB) as well as Electronic Code book (ECB) modes along with accomplish a 493Mbps of throughput. Parhi [30] applied s-box utilizing combinational logic to obtain optimum throughput. Furthermore, a fresh Key Expansion program was offered to function with various key lengths. It accomplished an optimum 21.56 Gbps of throughput with 168.4MHz frequency. Kermani and Masoleh [31] suggested a powerful approach to utilize s-box by reducing logic gates. Greater output and decrease time delay for sub-key procedure and equipment structures were offered. In the research by Jamal [32], a coalesced modification system was recommended through which various procedures were mixed with each other. The s-box variable and Mix Column matrix variables were additionally precomputed. Round keys had been furthermore precomputed prior to encryption procedure starts and were saved in RAM (RAM). The technique provided a 5.3 Gbps of throughput in Virtex-7 unit with 456MHz frequency. Rashidi [33] suggested a minimal power AES encryption regarding an image encryption application which employs a four-stage pipelined structures. Oukili [34] offered a pipelined structure for amalgamated field implementation of s-box along with key extension component. It could actually attain a optimum 108.69 Gbps throughput over a Virtex-6 unit. Oukili [35] suggested implementation of AES utilizing a 5-phase pipelined structures for s-box and also a 7-phase pipelined key extension. Furthermore, the side route attack is prevented by covering up the s-box. The initial reason for working on the suggested work was the necessity for higher -throughput AES execution for cloud conditions where encryption is actually a regular procedure. As the Xeon-FPGA system has been introduced for information stores by Intel, the running energy and versatility of FPGA need to be utilized for designing of a higher-throughput, lower -area accelerator. Secondly, the overall flexibility of FPGA equipment enables customized multistage pipelining for particular issues. Finally, a harmony among throughput and resources ought to be managed. Incrementing the throughput outcomes in the increment of resource usage in most the associated works. An effective technique ought to generate higher throughput along with fewer quantity of resources.

Whilst examining all the associated works, the pipelining effectiveness is enhanced by managing the phases by

allocating registers at suitable positions or by establishing the clock period to the time from the optimum time-taking procedure. These options will eliminate stalls. However, the optimum frequency that could be utilized cannot be improved above a particular limit as every pipelining phase needs a certain amount of time for finalization. In the suggested technique, we minimize this moment by memory space dividing and by implementing individual initiation interval sub-pipelining for every procedure right after the latency for a greater frequency is examined.

The primary efforts of the document are mentioned listed below:

• Memories partitioning is carried out, that enables reading and writing several input and output bits.

• Once examining the latencies of all segments, fresh multistage sub-pipelined structures are suggested for every module, that would make the initiation time period of all segments to a single for minimum feasible route delay. This may permit it in order to improve the frequency instead compared to establishing it to harmony the pipelining, that is applied in associated research works. The segments, that collectively requires latency of a single clock cycle, are usually joined to eliminate unwanted delays and assets.

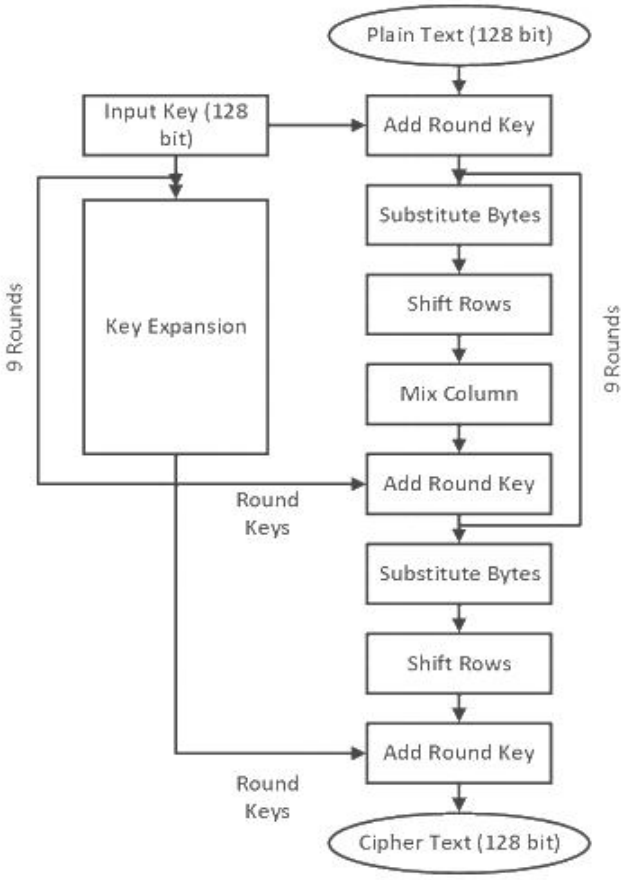

Figure 1: AES Algorithm

3. DESCRIPTION OF AES

AES is a symmetric blockchain encryption algorithm, its specified input length is 128 bits and key length is 128/119/256 bits in cycles of 10/12/14 depending on key length. The introduction of the AES has embraced the power of security, flexibility and efficiency. The AES basically contains four types of explicit text functions, namely Byte Swap (Byte Replacement), Shift Rows, Column Mix and Round Key Addition. The imported text is separated by 4 × 4 footnotes. The byte replacement function replaces each item in the status table with an item in the s-box. The S-box is made of GF (8) designed to protect against all attacks. Move lines the function moves the lines to the left without the first line. The Mix Column replaces each item in the status bar at a price based on all the values in this column. Adding a round key enables the XOR function of each case with a round key made up of a key combination function [36-39]. The performance of the NPP as a whole is shown in the figure. 1. Repeated use of the four functions and internal calculation creates AES safety features, namely randomization, confusion and distribution [40]. The AES installation pipes for each section are shown in the figure. 2. After each cycle, registers were added to maintain intermediate results and avoid multiple text conflicts.

4. PROPOSED METHOD

In the recommended structures, we evaluate establishing an effective method to enhance pipelining. One particular bottleneck happens throughout the entry of input information and key coming from memory space. The input information's are saved within BRAMs or LUTs within FPGA, that continues to be utilized by various procedures at various pipelining phases. In order to minimize the accessibility time and to create pipelining much more effective, the memory space wherever inputs are saved could be partitioned directly into various banks. Partitioning the memory space into various banks enables assigning of extra slots to it for creating the accessibility parallel and therefore growing the effectiveness of pipelining [41]. Within the suggested technique, plain text and key saving memories are divided into several banks, that permits accessing the material inside a single clock cycle. Additionally, the layout is designed to create the initiation period of all operations so that it can easily take input in each and every clock cycle and generate outcomes in each and every clock cycle as soon as the pipeline is complete. This could be achieved by creating the input to be examine parallel and sub-pipelining every operation of AES by including registers at suitable positions to stability the pipeline. Furthermore, a much better style ought to balance among the resources utilized and the throughput accomplished. Increasing throughput along with a huge quantity of resources reduces the performance of the design. Figure 4 demonstrates the general layout of the suggested technique. The particular input Plain text and key memory space are partitioned to permit reading and transferring of input details for processing within just a single clock period. As Shift Rows never take any extra clock cycle, it is combined with Substitute Byte, that is mentioned in earlier. The crucial

route is separated into several parts through allocating registers at suitable placements, permitting better pipelining.

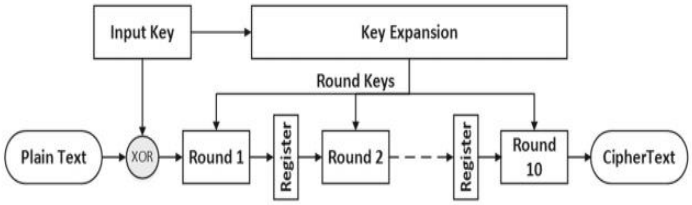

Figure 2: The implementation of AES pipelined

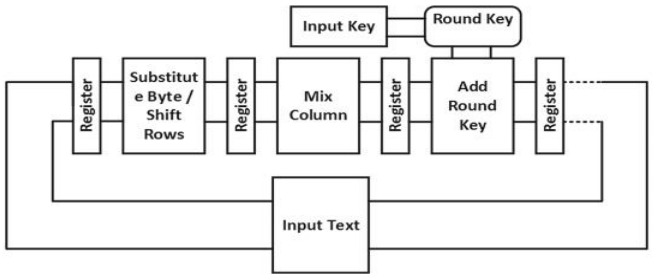

Figure 3: Suggested Cycle for Advanced Encryption Standard

The procedures are sub-pipelined within every procedure so that it might obtain input and move it to the subsequent phase of pipeline at every clock period. Which indicates the initiation period of every procedure and sub-operations is decreased to Just one. Substitute Byte is actually the initial pipeline step that scans information from the partitioned memory space, procedures it, and provides it to register for the following phase. Shift Rows, Mix Column, and Add Round Key get input information through register for handling, and therefore, each can go through information in a single clock period.

Add Round Key is definitely the final pipeline phase which reads information through register and key through memory space. The key storage space is additionally partitioned to permit parallel accessibility. Following this, Add Round Key creates data back again to input information memory that is previously partitioned. The significant factors which determine the performance of pipelining tend to be latency [41]. Latency of an instruction is identified as the quantity of clock cycles it requires to finish execution, which is, the number of process between an instruction required to procedure the information and to create the result accessible for the following instruction. Initiation time period specifies how frequently input could be provided to an instruction, which is, the quantity of cycles among giving a couple of directions of the exact same type. Initiation period establishes the throughput of the layout. If the latency is too higher, initiation time period will turn out to be higher and the quantity of stalls will improve. A fine pipelined pattern ought to have reduce latency for every functional unit as well as input should be provided at every clock period. If the latency of the procedure (with regard to the suggested pattern, we utilized procedure rather of instruction) cannot reduce, it ought to be sub-pipelined. A sub-pipelined style raises the pipelining performance as every procedure is separated directly into sub-operations, that enables information to be traversed by means of every substages and inhibits stalling [41,42]. The latency values with regard to various route time delay as well as frequencies noticed for the suggested method are demonstrated in Table 1. Various frequencies are chosen dependent on the delay essential for every sub-operation. With regard to the route 4.91 ns time delay, the latency of almost all procedures is Zero along with memory space partition. Nevertheless, for which route delay, the highest frequency value which can be utilized is restricted to 203 MHz. Whilst reducing the route delay, the latency of procedures begins to improve. The optimum frequency which could be accomplished is together with 1.23 ns route delay and well-balanced resources and time period. At this path delay i.e. 1.23 ns, as the latency of Key Expansion and Mix Column are 4 and 3, correspondingly, the initiation time period will be 5 for Key Expansion and Mix Column. Similarly, with regard to time delay of 2.6 ns, the initiation time period regarding Mix Column is going to be 2 and for Key Expansion is going to be 3. In conventional technique, the clock interval is established to the longest latency procedure, that decreases the throughput, losing time unnecessarily with regard to the procedures that usually do not require that a lot of time period.

Table 1: Latency values V/s Frequencies

S. No.	Path Delay (ns)	Freq. (MHz)	Latency			
			Substitute Byte Shift Rows	Mix Column	Add Round Key	Key Expansion
1	1.231	813.1	0	3	0	4
2	2.60	357.1	0	1	0	2
3	2.840	352.1	0	0	0	2
4	4.910	203.1	0	0	0	0

The solution is applying sub-pipeline within every procedure. The associated works state that managing the pipelining raises throughput. We have followed a different strategy. To produce excellent pipelining, the initiation time period of all procedures should be Just one. Whenever the initiation time period of all procedure becomes One, it may obtain input at every single clock cycle. The objective is accomplished in the perform by sub-pipelining each procedure by having registers at suitable positions. In some other scientific studies [14,16-18], the sub-pipelining is transported out by just managing the pipelining not targeted at producing the initiation interval to Just one.

5. IMPLEMENTATION AND RESULTS

The suggested structure is tested, simulated, and applied various FPGAs-XC5VLX85, XC7VX690T, and XC6VLX240T systems. We have utilized Xilinx ISE Suite for the execution [43-45]. The best possible resource utilization for maximum frequency is considered in this research work. Initially, memory segmentation is carried out and examined the latency necessity for various procedures.

The input and key keeping memories as well as the s-box memory space were segmented in order to enable simultaneous accessibility. The latency distinction noticed prior to and right after memory segmentation is demonstrated in Fig. 4 i.e. 1.23 ns. The other parameter values are demonstrated in Table 2. Table 2 demonstrates the latencies for various route delay and the initiation time period accomplished using suggested approaches. It is observed that the implementation is completely pipelined as well as balanced and tends to make the initiation interval of all procedures to be Just one. With regard to a route delay of 1.23 ns, Mix Column latency is 3. By utilizing three-phase pipelining, the initiation time period is decreased through 4 to 1, that enhances the pipelining and also throughput. For Key Expansion latency is 4 and the initiation time period is 1. We chose the route delay of 1.23 ns for acquiring optimum throughput. It is observed that a highest frequency of 813MHz was accomplished along with the initiation time period of each and every procedure kept as Just one clock period. Various sub-pipelining methods with various frequencies are examined and discovered that 1.23-ns route delay implementation provides maximum throughput along with a bit more resources in the type of registers compared to other route delays. Providing several inputs to the layout enables

performance of procedures in pipelined way along with an initiation time period of A single, as demonstrated in Fig. 6. The actual design is demonstrated in Fig. 5. Figure 6 exhibits the simulation overall performance view of performing a couple of input texts along with 1.23-ns route delay. Every cell signifies a single clock cycle [40]. It is observed that Substitute Byte and Shift Rows are carried out inside just one clock cycle. The initiation time period of almost all procedures is Just one, and it is used again within just a single clock cycle. Mix Column is using four clock time cycle and Key Expansion is using five clock time cycles in working procedures in a pipelined approach together with an initiation period of 1, that will not include any kind of delay. We have utilized several resources of the similar type to prevent structural threat. For clearness, we have demonstrated only a couple of inputs in Fig. 6. Because outcome of this, throughout every clock cycle, the actual input traverses through one phase to another with 813MHz without having leading to any delay for following input. The latency of the entire system doesn't influence the processing speed and performance can be determined as presented [14,17,35]. Quantity of outcome bits for the technique is 128-bits. Crucial route delay can be determined as clock time period × initiation time period. As the initiation time period of general program is 1, the latency will not impact the throughput of the program. So, the crucial route delay is going to be 1.23 ns. The suggested system accomplishes a throughput of 104.06 Gbps. The general latency of the encryption is 58-time clock periods for 128-bit input information. The comparison at optimum frequency which can be accomplished, and the related throughput and resources utilized in several associated works with our suggested work in Table 3. Whilst considering various units, it is discovered that Virtex5 utilizes 65-nm technologies; Virtex-6 utilizes 40-nm technologies; and Virtex-7 utilizes 28-nm technologies. The optimum frequency which can be provided is restricted by the type of unit Therefore, generally there will be a smaller variance in essential route delay. The cloud machine will be utilizing most recent Virtex-7 unit, and therefore, the layout of the suggested work had been focused to develop structures that is suitable for changing to greater frequency.

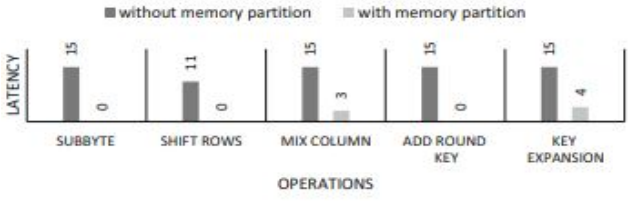

Figure 4: Latency requirement for 1.23-ns path delay at different memory partition

Table 2: Statistics of initiation interval and Latencies of pipelining operations for different path delays

S. No.	Path Delay (ns)	Freq. (MHz)	Interval				Latency			
			Sub Byte Shift Rows	Mix Column	Add Key	Key Ex	Sub Byte Shift Rows	Mix Column	Add Key	Key Ex
1	1.231	813	1	1	1	1	0	3	0	4
2	2.6	357	1	1	1	1	0	1	0	2
3	2.84	352	1	1	1	1	0	0	0	2
4	4.91	203	1	1	1	1	0	0	0	0

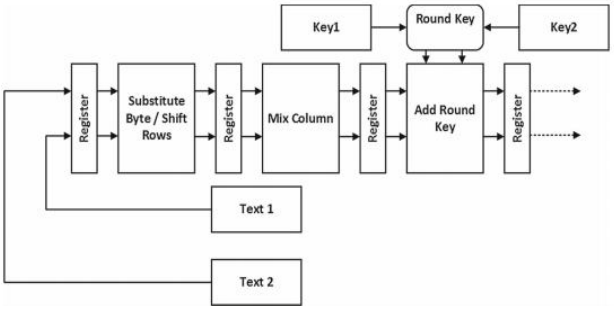

Figure 5. Suggested technique with multiple input texts

Figure 6: Performance analysis of two input text execution with 1.23-ns path delay

.

Table 3: Comparison studies of related works to the suggested design

Ref	Platform	Frequency (MHz)	No. of Bits	Path Delay (ns)	Slices	Throughput (Gpbs)	Efficiency (Mbps)
[19]	XC3S4000-5	240.900	128	-	20720	30.1120	**1.4100**
	XC2V8000-5	222.800	128	-	31674	27.8500	**0.9100**
[20]	XC2V6000	305.100	128	-	10662	38.1300	**3.6630**
[22]	XC5VLX30	277.400	128	3.6182	-	0.2700	-
[18]	XC5VLX330	555.000	128	3.1300	-	3.8000	-
[26]	XC7Z020-2CLG484	239.648	128	-	-	5.7510	-
[23]	XC7VX690T	516.800	128	-	3436	66.1000	**19.2000**
[24]	XC5VLX50T	90.440	128	-	-	11.5700	**7.0000**
[30]	XCV1000 e-8bg560	168.400	128	-	11022	21.5600	**1.9560**
[28]	XC2V6000-6	194.700	128	5.1360	3576	24.9220	**6.9000**
[16]	XC7VX690T	593.000	128	1.6000	4339	75.9200	**17.5000**
	XC6VLX240T	573.000	128	1.7000	3900	73.3900	**18.8100**
	XC5VSX240T	439.170	128	2.2000	4444	56.2100	**12.6500**
	XC5VLX110T	403.390	128	2.4000	4445	51.6300	**11.6200**
	XC4VLX160	454.550	128	2.1000	38511	58.1800	**1.5100**
[14]	XC5VLX85	433.060	128	-	3557	55.4320	**15.5800**
	XC5VLX85	528.370	128	-	3557	67.6310	**19.0100**
	XC5VLX85	671.524	128	-	3557	86.0000	**24.1800**
[17]	XC5VLX85	553.710	128	1.8000	10733	70.8700	**6.6000**
	XC5VLX85	554.785	128	1.8000	10352	71.0100	**6.8500**
	XC5VLX85	644.330	128	1.5000	28592	82.4700	**2.8800**
	XC6VLX240T	764.059	128	1.3000	10760	97.8000	**9.0800**
	XC6VLX240T	803.988	128	1.2000	28520	102.9100	**3.6000**
[32]	Virtex 7	456.000	128	2.1900	2444	5.3000	**2.1700**
[33]	Stratix II	475.000	128	2.1000	808 Registers	0.6025	-
[34]	XC5VLX85	638.162	128	1.5600	7385	81.6800	**11.0600**
	XC6VLX240T	849.185	128	1.1700	6361	108.6900	**17.0800**
[35]	XC6VLX240T	732.279	128	1.3000	5759	93.7300	**16.2700**
	XC6VLX240T	457.582	128	2.1000	9531	58.5700	**6.1400**
Our Deign	XC5VLX85	704.700	128	1.4200	2940	90.4000	**30.7400**
	XC6VLX240T	740.700	128	1.3500	2537	94.8100	**37.3700**
	XC7VX690T	**813.000**	**128**	**1.2300**	**2617**	**104.0600**	**39.7000**

The minimal critical route delay that can be accomplished for the suggested program on XC6VLX240T unit is 1.35 ns and this for XC5VLX85 unit is 1.42 ns that is demonstrated in Table 3. A great program must have a stability among the throughput and resource usage. The suggested system will be able to accomplish a greater throughput along with less reference by very carefully enhancing the source program code as well as resource suitable placement. As sub-pipelining of the style is carried out after examining the latency for greater frequency and route delay, improved throughput by appropriate managing might be accomplished. In this article by Farashahi et al. [14], the 2-slow time retiming approach obtained an optimum 82.47 Gbps throughput. Liu et al. [16] might get an optimum 75.92 Gbps throughput along with 593MHz clock frequency. The suggested system accomplished of 90.4 Gbps throughput was Virtex-5 unit, Virtex-6 throughput was 94.81 Gbps, and Virtex-7 throughput was 104.06 Gbps.

It is observed that actually if the suggested technique accomplishes fewer throughput compared to which in the research works completed by Bri [34] and Sharifian [17] on FPGA Virtex-6, the resource usage of the suggested technique is significantly much less, that results in greater efficiency. The suggested system is 27.13% effective when evaluating with the greatest performance demonstrated in the study doneon on FPGA Virtex 5 by Farashahi et al. [14]. The performance of the suggested work is identified to be far better than those reported in evaluation dependent on implementation of comparable units such as Virtex-6 as well as Virtex-5. The AES core is built-in utilizing higher-performance PCIe 3.0 user interface. Figure 7 indicates the highest frequencies attained by various earlier proposals and the suggested approach on different FPGA Units. For convenience, we have chosen only the earlier works displaying best efficiency. It is observed that the suggested work can accomplish highest throughput on Virtex-7 on 813MHz. Figure 8 displays the evaluation among throughput of associated works and the suggested work on various FPGA Units. Also, the evaluation of resource usages is demonstrated in Figure 9.

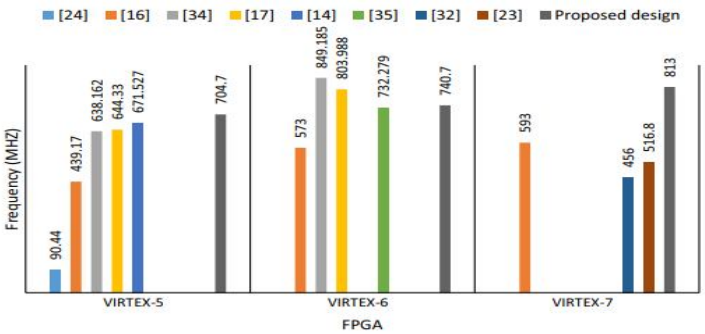

Figure 7: Comparative study of frequencies carried out on different FPGAs.

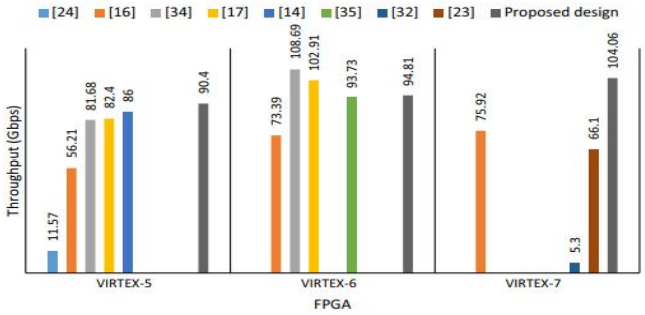

Figure 8: Comparison of throughputs achieved by different designs on different FPGAs.

The suggested technique is demonstrated to generate maximum throughput other than the style demonstrated by Bri [34] and Sharifian [17] on Virtex-6 unit. While whenever examining the resource usage, it is observed that the suggested technique is much more efficient than all additional approaches. Whilst putting it completely, the efficiency for the suggested model was observed to be 30.74Mbps, 37.37Mbps and 39.7Mbps on Virtx-5 Unit, Virtx-6 Unit and Virtex-7 FPGAs Unit, respectively (Fig. 10). Figure 11 demonstrates the evaluation of percent boost in efficiency of our suggested program when evaluating along with the best implementations on various boards.

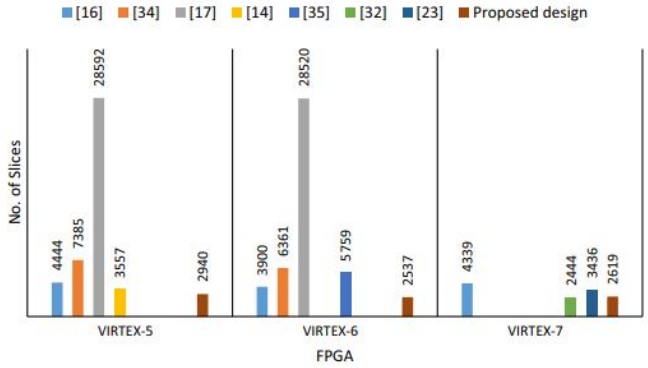

Figure 9: Comparison of number of slices (resource usage) used by different designs on different FPGAs.

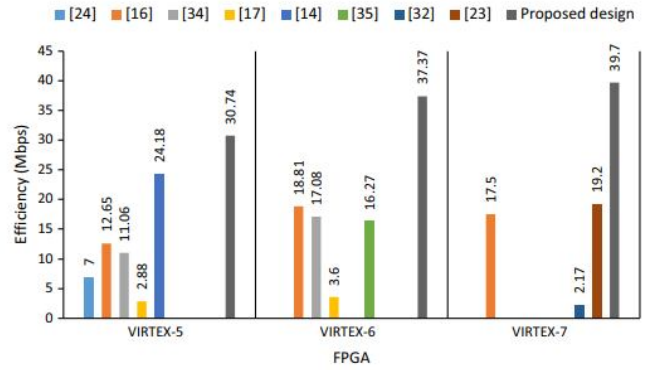

Figure 10: Comparison of efficiency of different designs on different FPGAs.

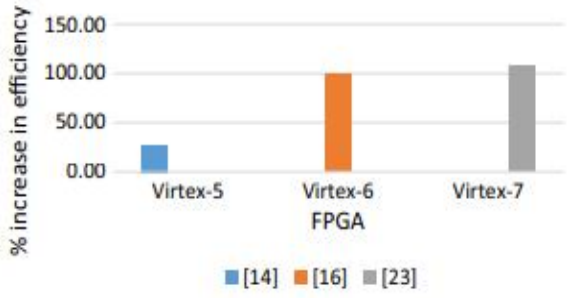

Figure 11: Efficiency attained by the suggested design implementations on different FPGA boards.

It is observed that actually if the suggested technique accomplishes fewer throughput compared to which in the research works completed by Bri [34] and Sharifian [17] on FPGA Virtex-6, the resource usage of the suggested technique is significantly much

less, that results in greater efficiency. The suggested system is 27.13% effective when evaluating with the greatest performance demonstrated in the study doneon on FPGA Virtex 5 by Farashahi et al. [14]. The performance of the suggested work is identified to be far better than those reported in evaluation dependent on implementation of comparable units such as Virtex-6 as well as Virtex-5. The AES core is built-in utilizing higher-performance PCIe 3.0 user interface. Figure 8 indicates the highest frequencies attained by various earlier proposals and the suggested approach on different FPGA Units. For convenience, we have chosen only the earlier works displaying best efficiency. It is observed that the suggested work can accomplish highest throughput on Virtex-7 on 813MHz. Figure 9 displays the evaluation among throughput of associated works and the suggested work on various FPGA Units. Also, the evaluation of resource usages is demonstrated in Fig. 10. The suggested technique is demonstrated to generate maximum throughput other than the style demonstrated by Bri [34] and Sharifian [17] on Virtex-6 unit. While whenever examining the resource usage, it is observed that the suggested technique is much more efficient than all additional approaches. Whilst putting it completely, the efficiency for the suggested model was observed to be 30.74Mbps, 37.37Mbps and 39.7Mbps on Virtx-5 Unit, Virtx-6 Unit and Virtex-7 FPGAs Unit, respectively (Fig. 11). Figure 12 demonstrates the evaluation of percent boost in efficiency of our suggested program when evaluating along with the best implementations on various boards.

Along with less area usage and higher throughput, the suggested AES accelerator is much better suitable for cloud application wherever encryption/decryption of huge amount of information is being managed. As FPGA resources are contributed as support on cloud, resource consumption of AES accelerator is of the same significance as throughput. Along with a throughput of 104.06 Gbps and area consumption of 2617 slices, it might meet the both speed necessity and optimum utilization of distributed assets in cloud. In brief, the suggested model could accomplish a much better efficiency of 39.7Mbps, 37.37Mbps, and 30.74Mbps on Virtex-7, Virtex-6 and Virtex-5FPGAs unit respectively.

6. CONCLUSION

A much better implementation of AES accelerator along with excellent multi-phase sub-pipelined design for a route time delay of 1.23 ns in XC7VX690T unit is suggested. The suggested technique could be utilized for applications wherever higher throughput is needed. Through memory space segmentation, sub-pipelining, pipelining, unrolling, and function-merging, the suggested approach outperforms all techniques reviewed in other research works. Examining the effect of higher frequency on functionality latency and memory space accessibility latency, we might layout a much more effective sub-pipelining approach. As the initiation time period of all procedures and the whole technique is A single, generally there will be absolutely no delay in acquiring the inbound information at the specific rate. The method can handle at a optimum frequency at 813MHz on the most recent XC7VX690T unit along with significantly fewer resource usage and might accomplish a 104.06 Gbps throughput. The exact same technique whenever examined on XC6VLX240T and XC5VLX85 units along with route delays of 1.351ns and 1.425 ns could generate a throughput of 94.810 Gbps and 90.461 Gbps, respectively. Performance of the suggested system outperforms that attained by the existing approaches.

REFERENCES

1. Seskar, I. ***Advanced FPGA Design***. IEEE Signal Processing Magazine, 25(6), 173-174. 2008 https://doi.org/10.1109/msp.2008.929815.
2. Phan, R. ***Impossible differential cryptanalysis of 7-round Advanced Encryption Standard (AES). Information Processing Letters***, 91(1), 2004, 33-38. https://doi.org/10.1016/j.ipl.2004.02.018.
3. Javeed, K., & Wang, X**.** ***FPGA Based High Speed SPA Resistant Elliptic Curve Scalar Multiplier Architecture. International Journal of Reconfigurable Computing***, 2016, 1-10. https://doi.org/10.1155/2016/6371403.
4. Teubner, J., & Woods, L. ***Data Processing on FPGAs. Synthesis Lectures on Data Management***, 5(2), 1-118. 2013 https://doi.org/10.2200/s00514ed1v01y201306dtm035.
5. Bokefode, J., Bhise, A., Satarkar, P., & Modani, D. ***Developing A Secure Cloud Storage System for Storing IoT Data by Applying Role Based Encryption. Procedia Computer Science***, 89, 43-50. 2016 https://doi.org/10.1016/j.procs.2016.06.007.
6. Rahmani, H., Sundararajan, E., Ali, Z., & Zin, A. ***Encryption as a Service (EaaS) as a Solution for Cryptography in Cloud. Procedia Technology***, 11, 1202-1210. 2013 https://doi.org/10.1016/j.protcy.2013.12.314.
7. Chu, C., Ouyang, Y., & Jang, C. **Secure data transmission with cloud computing in heterogeneous wireless networks. Security and Communication Networks, 5(12)**, 2012 1325-1336. https://doi.org/10.1002/sec.409.
8. Bankar, S. ***Cloud Computing Using Amazon Web Services AWS. International Journal of Trend In Scientific Research And Development***, 2018 Volume-2(Issue-4), 2156-2157. https://doi.org/10.31142/ijtsrd14583.
9. Microsoft Windows Azure: ***Developing Applications for Highly Available Storage of Cloud Service***. (2015), 4(12), 662-665. https://doi.org/10.21275/v4i12.nov151864.
10. ***Sentiment analysis on google cloud platform.*** (2020). https://doi.org/10.48009/2_iis_2020_221-228.
11. S, V., & T, P. ***An Efficient Securing Code Based Cloud Storage using RC5 Encryption Algorithm against Pollution Attacks. International Journal of Trend in Scientific Research and Development***,

2018 Volume-2(Issue-2), 1652-1657. https://doi.org/10.31142/ijtsrd10744.

12. Babb, B. ***Commentaries on the IAALS' Honoring Families Initiative White Paper. Family Court Review,*** 52(4), 639-641. 2014 https://doi.org/10.1111/fcre.12114.
13. Viswanath, G., & Krishna, P. ***Hybrid encryption framework for securing big data storage in multi-cloud environment. Evolutionary Intelligence.*** 2018 https://doi.org/10.1007/s12065-020-00404-w.
14. Farashahi, R., Rashidi, B., & Sayedi, S. ***FPGA based fast and high-throughput 2-slow retiming 128-bit AES encryption algorithm. Microelectronics Journal, 45(8), 1014-1025. 2014*** https://doi.org/10.1016/j.mejo.2014.05.004.
15. Jing, M., Chen, Z., Chen, J., & Chen, Y. ***Reconfigurable system for high-speed and diversified AES using FPGA. Microprocessors and Microsystems, 31(2),*** 2007, 94-102. https://doi.org/10.1016/j.micpro.2006.02.018.
16. Liu, Q., Xu, Z., & Yuan, Y**.** ***High throughput and secure advanced encryption standard on field programmable gate array with fine pipelining and enhanced key expansion. IET Computers & Digital Techniques,*** 2015, 9(3), 175-184. https://doi.org/10.1049/iet-cdt.2014.0101.
17. Soltani, A., & Sharifian, S. ***An ultra-high throughput and fully pipelined implementation of AES algorithm on FPGA. Microprocessors and Microsystems,*** 2015, 39(7), 480-493. https://doi.org/10.1016/j.micpro.2015.07.005.
18. Lee, H., Paik, Y., Jun, J., Han, Y., & Kim, S**.** ***High-throughput low-area design of AES using constant binary matrix-vector multiplication. Microprocessors and Microsystems,*** 2016 47, 360-368. https://doi.org/10.1016/j.micpro.2016.10.003.
19. Good, T., & Benaissa, M**.** ***Pipelined AES on FPGA with support for feedback modes (in a multi-channel environment). IET Information Security***, 2007, 1(1), 1. https://doi.org/10.1049/iet-ifs:20060059.
20. Hammad, I., El-Sankary, K., & El-Masry, E. ***High-Speed AES Encryptor With Efficient Merging Techniques.*** 2010 IEEE Embedded Systems Letters, 2(3), 67-71. https://doi.org/10.1109/les.2010.2052401.
21. Sugawara, T**.** ***3-Share Threshold Implementation of AES S-box without Fresh Randomness. IACR Transactions on Cryptographic Hardware and Embedded Systems,*** 2018 123-145. https://doi.org/10.46586/tches.v2019.i1.123-145.
22. Kalaiselvi, K., & Mangalam, H. ***Power efficient and high-performance VLSI architecture for AES algorithm. Journal of Electrical Systems And Information Technology, 2(2),*** 2015 178-183. https://doi.org/10.1016/j.jesit.2015.04.002.
23. P., R., & H., M**.** ***Design and implementation of power and area optimized AES architecture on FPGA for IoT application. Circuit World, ahead-of-print(ahead-of-print).*** 2020 https://doi.org/10.1108/cw-04-2019-0039.
24. Yang, C., & Chien, Y**.** ***FPGA Implementation and Design of a Hybrid Chaos-AES Color Image Encryption Algorithm. Symmetry, 12(2),*** 2020, 189. https://doi.org/10.3390/sym12020189.
25. Arul Murugan, C., Karthigaikumar, P., & Sathya Priya, S. ***FPGA implementation of hardware architecture with AES encryptor using sub-pipelined S-box techniques for compact applications. Automatika,*** 2020, 61(4), 682-693. https://doi.org/10.1080/00051144.2020.1816388.
26. Nabil, M., M. Khalaf, A., & M. Hassan, S. ***Design and implementation of pipelined and parallel AES encryption systems using FPGA***. 2020, Indonesian Journal of Electrical Engineering and Computer Science, 20(1), 287. https://doi.org/10.11591/ijeecs.v20.i1.pp287-299.
27. Thiyagarajan, K., El-Sankary, K., Wang, Y., & Hammad, I. ***Low Complexity Multimedia Encryption. International Journal of Computer Network and Information Security,*** 2016, 8(4), 1-13. https://doi.org/10.5815/ijcnis.2016.04.01
28. Granado-Criado, J., Vega-Rodríguez, M., Sánchez-Pérez, J., & Gómez-Pulido, J**.** ***Hardware security platform for multicast communications. Journal of Systems Architecture,*** 2014, 60(1), 11-21. https://doi.org/10.1016/j.sysarc.2013.11.007
29. Chiţu, C., & Glesner, M. ***An FPGA implementation of the AES-Rijndael in OCB/ECB modes of operation.*** 2005, Microelectronics Journal, 36(2), 139-146. https://doi.org/10.1016/j.mejo.2004.10.012
30. Xinmiao Zhang, & Parhi, K. ***High-speed VLSI architectures for the AES algorithm. IEEE Transactions on Very Large-Scale Integration (VLSI) Systems,*** 2005, 12(9), 957-967. https://doi.org/10.1109/tvlsi.2004.832943
31. Mozaffari-Kermani, M., & Reyhani-Masoleh, A. ***Efficient and High-Performance Parallel Hardware Architectures for the AES-GCM. IEEE Transactions on Computers***, 2005, 61(8), 1165-1178. https://doi.org/10.1109/tc.2011.125
32. Zhao, J., Guo, Z., & Zeng, X., ***High Throughput Implementation of SMS4 on FPGA***. 2019, IEEE Access, 7, 88836-88844. https://doi.org/10.1109/access.2019.2923440.
33. Hamidi, I., & Al-aassi, F. ***High Speed FPGA Based 128-bit Advance Encryption Standard (AES). International Journal Of Sensors, Wireless Communications And Control,*** 2015, 11. https://doi.org/10.2174/2210327911666210201104151.

34. Oukili, S., & Bri, S., ***High throughput FPGA Implementation of Advanced Encryption Standard Algorithm. TELKOMNIKA (Telecommunication Computing Electronics And Control),*** 2017, 15(1), 494. https://doi.org/10.12928/telkomnika.v15i1.4713.
35. Oukili, S., & Bri, S***., Hardware Implementation of AES Algorithm with Logic S-box. Journal Of Circuits, Systems And Computers, 26(09),*** 2017, 1750141. https://doi.org/10.1142/s0218126617501419.
36. Hussien, M***. Encryption of Stereo Images after Compression by Advanced Encryption Standard (AES). Al-Mustansiriyah Journal of Science, 28(2),*** 2017, 156. https://doi.org/10.23851/mjs.v28i2.511.
37. Silva, L., B. P, D., & Heriyanto, H. ***Aplikasi enkripsi dan dekripsi file dengan menggunakan aes (advanced encryption standard) algoritma rijndael pada sistem operasi android. Telematika,*** 10(1). 2015, https://doi.org/10.31315/telematika.v10i1.383.
38. Baux, K. Rankin SH, Stallings KD. Patient Education. The Journal of Cardiovascular Nursing, 5(3), 86. 1991. https://doi.org/10.1097/00005082-199104000-00012.
39. Heron, S. ***Advanced Encryption Standard (AES). Network Security***, 2009 (12), 8-12. https://doi.org/10.1016/s1353-4858(10)70006-4.
40. Wright, M. ***The Advanced Encryption Standard. Network Security,*** 2001(10), 11-13. https://doi.org/10.1016/s1353-4858(01)01018-2.
41. Hennessy, J., & Patterson, D. ***A new golden age for computer architecture. Communications of the ACM,*** 2019, 62(2), 48-60. https://doi.org/10.1145/3282307.
42. Obaidat, M. Book Reviews: ***ADVANCED COMPUTER ARCHITECTURE: Parallelism, Scalability, and Programmability by Kai Hwang McGraw-Hill, New York, N.Y., 1993***. 770 pages, Price: $56.95, ISBN: 0-07-031622-8. SIMULATION, 61(4), 250-250. https://doi.org/10.1177/003754979306100406.
43. ***An Efficient VLSI Design of 32X32 bit Multiplier using Wallace Tree Algorithm in Vivado HLS and Xilinx ISE Software using VHDL.*** (2020), 9(7), 490-495. https://doi.org/10.35940/ijitee.g5299.059720.
44. Shashidhara, K., & Srinivasaiah, H. ***Hardware co-simulation of 1024-point FFT and its Implementation, in Simulink, Xilinx Vivado IDE on Zynq-7000 FPGA.*** 2019, European Journal of Engineering Research and Science, 4(9), 58-64. https://doi.org/10.24018/ejers.2019.4.9.1501.
45. ***Accelerator Design for Ethernet and HDMI IP Systems for IoT using Xilinx Vivado 18.X. (2019),*** 8(10), 652-656. https://doi.org/10.35940/ijitee.j8786.0881019.