

IMPLEMENTATION FPGA BASED IMPLEMENTATION OF 128-BIT AES ALGORITHM USING VHDL

ATTAR MEHARAJ BANU¹, SYED NOORULLAH²

¹DECS 202T1D3801, ASHOKA WOMENS ENGINEERING COLLEGE, KURNOOL, A.P.

²Assistant Professor – ASHOKA WOMENS ENGINEERING COLLEGE, KURNOOL, A.P

ABSTRACT

In current world of computations, data encryption is of prominent importance. Many algorithms were developed for data encryption and decryption to prevent hacking. The Advanced Encryption Standard (AES) is one of the data encryption techniques. Two famous kinds of hardware implementation techniques are pipelining and loop-unrolling techniques. In pipelining, registers are inserted between each combinational processing element so that each input data block can be processed simultaneously in each processing element. In this work, a pipelined implementation of AES encryption algorithm is developed. The number of rounds of AES-128 encryption is 10 and an architecture implementing this cipher is called fully pipelined, when all data blocks of 10 rounds can be processed simultaneously. In the loop-unrolling technique one or multiple rounds of the algorithm are processed in the same clock cycle. Here only one round of the algorithm is implemented as a combinational processing element and a data register is also used to store the result obtained in the previous clock cycle. In this work, a128-bit AES is implemented and for each round of AES encryption, a different sub-key is used as the round key, which is produced by the key schedule algorithm based on the loop-unrolled technique, to produce the required sub-key for each round is done. The no of I/O, Slices in the proposed work are 386, 229 respectively with Spartan6 fpga with a minimum period of 5.813ns, maximum Frequency of 172.031MHz, minimum input arrival time before clock of 4.823ns, maximum output required time after clock of 5.588ns, throughput of 2.2Gbps. The no of I/O, Slices are 387, 264 respectively with Artix7 fpga with a minimum period of 3.397ns, maximum Frequency of 294.366MHz, minimum input arrival time before clock of 1.649ns, maximum output required time after clock of 1.669ns, throughput of 3.77Gbps.

Key Words - 128-bit AES, Encryption, Decryption, Pipelining, Loop-Unrolling.

1. Introduction

The Security is one of the biggest concerns in the developing world. It is important to ensure a safe transfer of information between communicating parties, protecting them from attacks. Many standards and developed encryption protocols are avail- able as resources and are used based on the requirements. In this thesis, we propose a customized encryption algorithm and an authentication scheme to safely transfer information. The algorithm is a variation of Advanced Encryption Standard (AES) and is carried out between multiple devices. AES uses only one private key (sym- metric key) to encrypt the data. Cryptography is the science of information security. Cryptography includes techniques such as microdots, merging words with images and other ways to hide information in storage or transit. Cryptography is most often associated with scrambling plaintext (ordinary text, sometimes referred to as clear text) into ciphertext (a process called encryption), then back again (known as decryption). A Cryptographic system that uses two keys – a public key known to everyone and a private or secret key known only to the recipient of the message. Individuals who practice this field are known as cryptographers. Symmetric cryptography can be split into block ciphers and stream ciphers. Fig. 1depicts the operational differences of a block cipher and stream cipher. Bits are encrypted individually in stream ciphers. A bit from a key stream is added to a plain text to achieve this. An entire block of plaintext is encrypted with the same key in block ciphers.



The encryption of the plaintexts in any given block is related to the encryption of another plaintext of the same block. Practically, the majority of block ciphers have a block length of 128 bits(16 bytes) like Advanced Encryption Standard (AES), or a length of 64 bits (8 bytes) like Data Encryption Standard (DES) or Triple DES (3DES) algorithm. AES and DES are examples of the symmetric algorithm. But, these are some of the many symmetric algorithms. Hundreds of algorithms have been proposed over the years. Even though some of these proposed algorithms were deemed insecure, many other cryptographically secure ones exist in the market.



Figure 1: Principles of encrypting n bits with stream and block ciphers

Advanced Encryption Standard (AES) is the prominent choice of algorithm for en- cryption which will be discussed further on. The AES cipher is similar to block cipher Rijndael. Rijndael with a block length of 128 bits is known as the AES algorithm.



AES has survived decades of brute force attacks with its three high key lengths of 128, 192 and 256 bits shown in Fig. 2. Any analytical attacks with a reasonable chance of success are still unknown. AES was a result of an open competition. There were four other potential strong finalist algorithms, Mars, RC6, Serpent, and Twofish. These were cryptographically strong and quite fast, particularly in software. Mars, Serpent, and Twofish are royalty-free. Triple DES often denoted as 3DES is an alternate to AES or the AES finalist algorithms. Three subsequent DES encryption with different keys are present in 3DES. 3DES does not support software as efficiently as it supports hardware. Financial applications and protection of biometric information in electronic passports use 3DES. The DES is made much more resistant against exhaustive key searches by this simple modification.

2. Literature Survey

Palm The data or information may be in any form such as text, audio, image, video, or others, making the data unreadable to attackers (Suresh & Ajai, 2016). Data encryption (Mishra et al., 2019) is done by encryption methods and the reverse of encryption is defined as decryption. To replace the data encryption standard (DES) for advanced technologies implementation, the advanced encryption standard (AES) method is proposed. There are special optimization techniques also followed in this hardware implementation, like lookup tables, pipelining (Swetha et al., 2017), etc. Alternatively, FPGA is mainly known for its security, high speed, flexibility, and low maintenance. These implementations are mainly used in block ciphers (Pandey et al., 2018). The AES algorithm analyzed in this review has been implemented in VLSI architecture to provide security and fast processing (Hameed et al., 2018). The AES algorithm is a symmetric encryption algorithm of cryptography. It is a block cipher algorithm discovered by Joan Daemen and Vincent Rijimen (Landge & Mishra, 2016). The AES 128-bit functions on a 4 4 matrix form array of array of bytes and size of block referred as a state. In this state, the encryption and decryption process is defined



in this state (Abdullah, 2017). This process is iterated in a number of rounds based on AES algorithm key length (number of bits in key) and mainly is used to transmit clear data into unclear data. Ten rounds of repetition used in 128-bit keys, for 192 -bit keys, 12 rounds, and for 256-bit keys, 14 rounds are used by AES based on key size. Multiple rounds of AES are used in hardware implementation of VLSI architecture. Each round is comprised of sub-bytes, shift rows, mix columns, and add round key (Siddesh & Shruthi, 2017).

3. Implementation

The proposed system of implementing 128-bit AES algorithm using HDL. Fig. 3 describes the structure of AES. It comprises of three layers, Key Addition layer, Byte Substitution Layer and Diffusion Layers (Shiftrow and Mixcolumn) respectively. Each layer manipulates 128 bits of the data. Before going through the process of how AES converts the plain text into cipher text, one has to know the standard properties exhibited by the AES fields, which are used in every layer of the algorithm. AES implementations are not limited to software applications. It is also used in hardware implementations such as FPGA's



Figure 3. Structure of AES.

An overview of a pipelined implementation of AES encryption algorithm is depicted in the following figure 4, where the round-i depicts the ith round of AES encryption algorithm.







International Journal of Engineering Technology and Management Sciences Website: ijetms.in Issue: 1 Volume No.7 January - February – 2023 DOI:10.46647/ijetms.2023.v07i01.019 ISSN: 2581-4621

The number of rounds of AES-128 encryption is 10 and an architecture implementing this cipher is called fully pipelined, when all data blocks of 10 rounds can be processed simultaneously. For a fully pipelined implementation of AES-128 ten 128-bit data registers are needed. The more data block we want to process simultaneously, the more registers and therefore, the more area we need for implementation. In contrast to pipelining, in a loop-unrolling technique one or multiple rounds of the algorithm are processed in the same clock cycle. In the smallest case of a loop-unrolled implementation of AES which is depicted in the following figure 5, only one round of the algorithm is implemented as a combinational processing element and a data register is also used to store the result obtained in the previous clock cycle.



Figure 5 Loop-Unrolled architecture of AES encryption algorithm.

For each round of AES encryption, a different sub-key is used as the round key, which is produced by the keyschedule algorithm. The following figure 6 represents one round of keyschedule algorithm.



Figure 6. Process of one round of key-schedule algorithm.

If someone wants to use a fixed key, it is preferable to calculate all sub-keys once and use a lookup table to store sub-keys, instead of implementing keyschedule algorithm and recalculate the sub-keys frequently. This strategy is especially suitable for software implementations, where memory is not as constrained as hardware implementations. The keyschedule is implemented based on the loop-unrolled technique to calculate the sub-keys on the fly. In other words, there is a dedicated part implementing keyschedule algorithm based on the loop-unrolled technique, to produce the required sub-key for each round on the fly. It is shown in below figure 7.





Figure 7 Architecure view of algorithm implemented.

The byte value in AES is represented as a set of bits (0 or 1) and is represented as the collection of bits separated by comma as {b7, b6, b5, b4, b3, b2, b1, b0}. These bytes are interpreted as finite field elements using polynomial representation as

(1)

b7x7 + b6x6 + b5x5 + b4x4 + b3x3 + b2x2 + b1x + b0

All the operations performed in AES are modulo-2 operations. These Operations are not as the same operations used in general Number System. The basic operations on which the entire math of the AES algorithm is based are Addition, Multiplication.

4. RESULTS

In order to implement 128-bit AES algorithm, keys we used are as listed below for plaintext: x"2a179373117e3de9969f402ee2bec16b",

key: x"3c4fcf098815f7aba6d2ae2816157e2b" ciphertext:

x"97ef6624f3ca9ea860367a0db47bd73a", The timing waveforms are shown in below figures

					110.1/5 ns
Name Value	0 ns	20 ns 40	0 ns 160 ns	80 ns 100 ns	
Vie clk 0					
la rst o					i
▶ 📲 key[127:0] 00000000000000000000000000000000000	000		3c4fcf098815f7aba6d2ae2816157e2b		000000000
plaintext[127:0]	000		340737e0a29831318d305a88a8f64332		000000000)
U done 1					
ciphertext[127:0] 320b6a19978511dcfb09dc021d84	39 UU (0848f8e92) 4950	16a024X9a463268dXe7585fd6	63 <mark>01beb8855X0cd5b\$5d3</mark> X258bd2fda)	0c048c7a6Xc5b09685aXd242c31beX320bt	a 199 (000)
🔓 clk_period 10000 ps			10000 ps		
reg_input[127:0] 00000000000000000000000000000000000	000 340737e0 4c2 e5b	0113b35	5e \b0c04c4c8 \d818d237e \43d82a34d \	(ea991bfac) bc429f4ca) 940709af5) 526	000000000
reg_output[127:0] 940709af5f892e3d722c32cbb570	Le9 UU\340737e0a\4c26	506287) e5b06b1ba) dcd025bb	b7)0113b35e6)b0c04c4c8)d818d237e)	(43d82a34)(ea991bfac)(bc429f4ca)(9407	09af5)000)
subbox_input[127:0] 320b6a19978511dcfb09dc021d84	539 UU\0848f8e92\4950	06a024) 9a463268d) e7585 d6	63)(01beb8855)(0cd5b\$5d3)(258bd2fda)	(0c048c7a6)c5b09685a)d242c31be)320bt	6 <mark>199)</mark> 000)
subbox_output[127:0] 232b02d4889782860f018677a455	512 UU	302771)b85a2345b) 946ac f6	50 \7cae6c975 \ fe03d\$4c2 \ 3f3db554d \	(fef264da4)a6e790979)(b52c2eaf9)(232b	0 <mark>2.d48)</mark> 636)
shiftrows_output[127:0] 88013fd40f5f0286a42b8277239	512 UU e598271ef 1a9	6de77f)b5117345d)075e50f6	6d\53fb4f97b\2329684cd\d3a9ab54f)	(4d8688dac)(95c3ec97d)(940709af5)(8801)	3 <mark>d40)</mark> 636)
mixcol_output[127:0] 526a217bd22b6c416354b1fc61c4	101 UU\4c2606287)e5b0	0113b35/06015bb7	5e \b0c04c4c8 \d818d237e \43d82a34d \	(ea991bfac) bc429f4ca (d70f20cd5) 526a	2.7bd)636)
Feedback[127:0] 526a217bd22b6c416354b1fc61c4	101 UU\4c2606287\e5b(0113b35/06015bb7	5e \b0c04c4c8 \d818d237e\43d82a34d)	(ea991bfac) bc429f4ca) 940709af5) 526a	2.7bd)636)
Found_key[127:0] a60c63b6c80c3fe18925eec9a8ff	1d0 UU\3c4fcf0988\057	56c2a3 X 7ff659737 X 3b887a6	5d \00ad0bdb3\bc15f911b \fd9300ca4)	(4fdca64eb)(2f298d7f6)(6e005c574)(a60ct	5 <mark>66c)</mark> 000)
▶ 📲 round_const[7:0] 6a	UU 01 (02 (04) 08	<u>(10)</u> (20) (40)	(<u>80) (16) (36) (</u>	6 01
Ug sel 0					
	V1, 110, 175 mg				
	X1: 110, 175 HS				





DOI:10.46647/ijetms.2023.v07i01.019 ISSN: 2581-4621

																		1	10.207 ns
Name	Value	0 ns			20 ns			40 ns		6	0 ns		80	ns			100 ns		
Ug cik 0	D																		
Ve rst o	D																		
ciphertext[127:0] 2	2e2b34ca59fa4c883b2c8aefd44be966							320	o6a19978511d	cfb09dc02	21d842	539							e2b34ca5
▶ 🛃 dec_key[127:0] 8	Bel88f6fcf5le923lle2923ecb5befb4		a60c63b6c80c3fe18925ee9a8f914d0												<u>ال</u>	e 188f6fcf			
lo done 1	1																		
plaintext[127:0] 3	340737e0a29831318d305a88a8f64332	(UU)	940709af5	bc429	4ca	ea991bfac	43d82	a34d	d818d237e	Xb0c04c4	k8 X	0113b35e6	dcd025bb	7 <mark>)</mark> e5	b06b1ba)	4c2606	287 X	340737e	0a <mark>X</mark> a03
Clk_period 1	10000 ps									10000	ps								
mux_output[127:0] 2	2e2b34ca59fa4c883b2c8aefd44be966	320b6a	a 199) d24	c5b096	85a	0c048c7a6	258bd	2fda	Ocd5b55d3	0 1beb8	355)	e7585fd63	9a463268	d)(49	506a024	0848f8	e92 🕅	5cc 2	e2b34ca5
reg_output[127:0]	0848f8e92a8dc69a2be2f4a0bee33d19	(UU)	320b6a199	d242c3	1be	c5b09685a	0c048	c7a6	258bd2fda	Ocd5b5	id3)	0 1beb8855	e7585fd6	3 <mark>)</mark> 9a	463268d>	(49506a	024)	0848f8e	92)2e2
inv_mixcol_input[127:0] 3	340737e0a29831318d305a88a8f64332	UU)	940709af5	bc429	4ca	ea991bfac	43d82	a34d	d818d237e	b0c04c4	k8)	0113b35e6	dcd025bb	7)e5	b06b 1ba	4c2606	287	340737e	0a)a03)
inv_mixcol_output[127:0]	De55de61148ff352ccbdc5db4a3bffal	UU	262ad1e8f	95c3ec	97d	4d8683dac	d3a9a	o54f	2329634cd	53fb49	7b)	075e50f6d	b5117345	d) 1a	96de77f	e59827	1ef)	0e55de6	11) (4fe)
invsr_input[127:0]	De55de61148ff352ccbdc5db4a3bffa1	UU)	940709af5	95c3e	97d	4d8683dac	d3a9a	o54f	2329634cd	53fb4f9	7b)	075e50f6d	b5117845	d) 1a	96de77f	e59827	(1ef)	0e55de6	11)a03
invsb_input[127:0] 4	4abdf3610e3bc5521455ffdbcc8fdeal	UU)	b52c2eaf9	a6e790	979	fef264da4	3f3db	554d	fe03d54c2	7cae6c9	75	946acff60	b85a2845	b) 3b	5302771	30524	1ee)	4abdf36	10 1fc
feedback[127:0]	5ccd7ed8d74907489bed7d9f27739cf1	UU)	d242c31be	c5b096	85a	0c048c7a6	258bd	2fda	0cd5b55d3	01beb8	355)	e7585fd63	9a463268	d)(49	506a024	0848f8	e92 🛛	5ccd7ed	8d (cbe)
Found_key[127:0] 3	3c4fcf098815f7aba6d2ae2816157e2b	(UU)	a60c63b6c	6e005	574	2f298d7f6	4fdcat	4eb	fd9300ca4	bc15f91	1b)	00ad0bdb3	3b887a6d	4)(7f	f659737)	05766c	2a3)	3c4fcf(9	88)8e1
Found_const[7:0]	00		36	X 1)	80	4	0	20	X 10		08	X 04		02	(01		00	36
is_first_round 0	D																		
		X1: 11	0.207 ns																

Figure 9 AES_DEC Simulated waveform



Figure 10. RTL view of AES_ENC.



Figure 11. RTL schematic view of AES_ENC.



International Journal of Engineering Technology and Management Sciences Website: ijetms.in Issue: 1 Volume No.7 January - February - 2023

DOI:10.46647/ijetms.2023.v07i01.019 ISSN: 2581-4621

□ ♪ ■ 単 🌡 X 0 0 × 10 0 × アクタタク 🗟 💌 🗣	1 🖬 🖬 🎤 🌾 🕨 🗵	*						
Design ↔ □ ₽ × 🕜 🕒 Design Overview	^	A	ES_ENC Project Status (09)/22/2022 - 11:30:33)				^
View: Vi	Project File:	Amehraj202T1	03801.xise P	arser Errors:		No Errors		
Hierarchy Module Level Utilization	Module Name:	AES_DEC	I	nplementation State:		Synthesized		
Amehraj2021103801	Target Device:	xc6slx16-2csg3	24	• Errors:				
AES_DEC - Behavioral (AEC_DEC	Product Version:	Product Version: ISE 14.7 • Warnings:						
AES_ENC - Behavioral (aes_enc.vhc 🔤 🔤 Static Timing	Design Goal:	Balanced		 Routing Results: 				
3	Design Strategy:	Xilinx Default (u	nlocked)	• Timing Constrain	its:			
Synthesis Messages	Environment:	System Setting	<u>s</u>	• Final Timing Scor	re:			
Translation Messages								
Place and Route Messages		Device	Itilization Summany (esti	mated values)			[.]	
< Timing Messages	Logic Utilization	Device	liced	Available	Utilizat	tion		
Running: Synthesis	Number of Sice Registers		194	44	18224	uon	10%	
Detailed Reports	Number of Sice LLTs		96	36	9112		106%	
Processes: Acs_DEC - Benavioral Synthesis Report Synthesis Report	V Number of fully used LLT-FF	nairs	130	7	10233		13%	
Design Properties	Number of bonded IOBs	pano	3	37	232		166%	
🕮 🗿 User Constraints 📃 🗌 Enable Message Filtering	Number of Block PAM/ETEO			16	32		50%	
View RTL Schematic	Number of BLEG/BLEGCTRL	,		1	16		6%	
View Technology Schematic Show Failing Constraints				*	10			
Check Syntax Show From Show From								
Implement Design			Detailed Reports				E	
Generate Programming File	Report Name	Status	Generated	Errors	Warnings	Infos		~
Start 🕫 Design 🚺 Files 🚺 Libraries 🗵 Design Summary (running)		aes_end	.vhd	×	1	1		
Console							⇔⊓é	×
8-bit xor4 : 16								^
								V

Figure 12. Design Summary of AES_ENC.







Figure 14. RTL schematic VIEW of AES_DEC.



International Journal of Engineering Technology and Management Sciences

Website: ijetms.in Issue: 1 Volume No.7 January - February – 2023 DOI:10.46647/ijetms.2023.v07i01.019 ISSN: 2581-4621

JSE Project Navigator (P.20131013) - C:\WINDOWS\system32\Amehraj202TID3801\Amehraj202TID3801\amehraj202TID3801.	xise - [Design Summary]					-	o x	×
	T 🕞 🎤 🚱 🕨 🗵 Ք	0						^
Design Overview ∧		<u>.</u>	AES DEC Pro	ject Status				٨
Image: Wew: ● 御 Implementation ○ Image: Simulation	Project File:	Amehraj202T1D3	3801.xise Pa	rser Errors:		No Errors		
I Hierarchy	Module Name:	AES_DEC	In	plementation Sta	te:	Synthesized		
Amehraj202T1D3801	Target Device:	xc6slx16-2csg324	4	•Errors:		No Errors		
- Negative Cogate - Negative Cogate - Negative Cogate - Negative Cogate - Clock Report	Product Version:	ISE 14.7		• Warnings:		2 Warnings (1 new)		
💻 🔤 AES_ENC - Behavioral (aes_enc.vho	Design Goal:	Balanced		 Routing Result 	lts:			
Salar Sa	Design Strategy:	Xilinx Default (un	locked)	• Timing Const	raints:			
🕼 🔛 👔 Synthesis Messages	Environment:	System Settings		• Final Timing S	icore:			
Iranslation Messages								
Place and Route Messages		Device Utilization Summary (estimated values)						
Timing Messages	Logic Utilization		AES_DEC Project Status tehraj202TID3801.xise Parser Errors: S_DEC Implementation State: SpEC Implementation State: anced • Routing Results: nx: Default (unlocked) • Timing Constraints: stem Settings • Final Timing Score: Device Utilization Summary (estimated values) Used Available 1944 18 9666 9 1397 10 387 1 Detailed Reports Status Detailed Reports Status Status Status Image: Status Image: Status Image: Status Image: Status		Uti	ilization		
Ko Processes Running	Number of Slice Registers			1944	18224		10%	
Processes: AES_DEC - Behavioral	Number of Slice LUTs		9	9686	9112		106%	
Design Summary/Reports	Number of fully used LUT-FF pairs			1397	10233		13%	1
Design Utilities Design Properties Design Constraint Design Properties	Number of bonded IOBs			387	232		166%	
Ger Constraints G	Number of Block RAM/FIFO			16	32		50%	
Implement Design Show Clock Report	Number of BUFG/BUFGCTRLs			1	16		6%	
Generate Programming File Snow Failing Constraints Software Target Device Show Warnings								
Analyze Design Using ChipScope			Detailed Report	5			[-]	
	Report Name	Status	Generated	Errors	Warnings	Infos		
	Construction Descent	- In	Thursday on the second day	0	5 Wanting 74 au		·	۷
Start Casign Control Files Control Libraries End Control Co	ımary	<u> </u>						
Console							↔□₽	X
Launching Design Summary/Report Viewer ▲ WARNING:ProjectMgmt - File C:/WINDOWS/system32/Amehraj202T1D3801/Dect ▲ WARNING:ProjectMgmt - File C:/WINDOWS/system32/Amehraj202T1D3801/Dect	ryption_process.stx is ryption_process.stx is	missing. missing.						^
4							>	۲
🗐 Console 🔇 Errors 🔔 Warnings 🙀 Find in Files Results								



In the reference [10] the number of slice register used is 954. In reference [11] researchers used two FPGA for implementing the AES design, for Virtex-5 FPGA the number of slice register used is 255. Below tables represent the comparison of existing system and proposed system results.

Resources	Used	Available
SR	3987	126800
LUT	4115	63400
I/O	269	300
BUFG	1	32

As seen in table-1, the existing work requires 4115 LUTs for the implementation of the AES algorithm. But for proposed work shown table-2 it is 1397 LUTs which effective utilization.

Table2: I	Proposed Sy	ystem.
Resources	Used	Available
SR	264	93296
LUT	1104	46648
I/O	387	408
BUFG	1	16

From the two tables the proposed system is occupying lesser area when compared to existing system 50% of area. With this result we can infer that the proposed one is much better than the existing system.



Website: ijetms.in Issue: 1 Volume No.7 January - February – 2023 DOI:10.46647/ijetms.2023.v07i01.019 ISSN: 2581-4621

																				110.17	<mark>5 ns</mark>
Name	Value	0 ns			20	ns			40 ns			60 ns			80 ns			100 ns			
U ₀ clk	0																				
U _e rst	0																				
🕨 💑 key[127:0]	000000000000000000000000000000000000000								3c4	fcf098815f7ab	a6d2ae2	81615	7e2b							000000	000
Interster [127:0]	000000000000000000000000000000000000000								340	737e0a298313	18d305a	88a8f6	54332							000000	000
U _o done	1																				
ciphertext[127:0]	320b6a19978511dcfb09dc021d842539	UU	0848f8e	92 <mark>)</mark> 49	506a02	24	a463268d	e7585	d63	01beb8855	Ocd5b	5 5d 3	258bd2fda	0c0480	. 7 a6	c5b09685a	d242c	1be)	320b6a	199)	000)
🖟 clk_period	10000 ps										1000	0 ps									
▶ 📲 reg_input[127:0]	000000000000000000000000000000000000000	3407	37e0 X4	c2/e5	b06b 1b	aXo	cd025bb7	0113	35e	b0c04c4c8	d818d	237e	43d82a34d	ea991	bfac	bc429f4ca	<mark>94070</mark>	af5	526	000000	000
reg_output[127:0]	940709af5f892e3d722c32cbb57d31e9	UU	340737e	0a <mark>/</mark> 4c	260628	37) (e	5b06b1ba	dcd02	bb7	0113b35e6	b0c04	c4c8	d818d237e	43d82	a34	ea991bfac	bc429	4ca	94070	af5)	000
subbox_input[127:0]	320b6a19978511dcfb09dc021d842539	UU	0848f8e	92 \ 49	506a02	24)(9	a463268d	e7585	d63	01beb8855	Ocd5b	55d3	258bd2fda	0c048	. 7 a6	c5b09685a	d242c	1be)	320b6a	199)	000
subbox_output[127:0]	232b02d4889782860f018677a45f3f12	UU	3052411	ee)(3b	530277	71)(b)85a2345b	946ac	f60	7cae6c975	fe03d	54c2	3f3db554d	fef264	da4	a6e790979	b52c2	:af9	232b0:	d48)	636
shiftrows_output[127:0]	88013fd40f5f0286a42b827723978612	UU	e598271	.ef 🛛 1a	196de77	7f) (b	5117345d	075e5	0f6d	53fb4f97b	23296	34cd	d3a9ab54f	4d868	8dac	95c3ec97d	94070)af5	88013	d40)	636
mixcol_output[127:0]	526a217bd22b6c416354b1fc61c48401	UU	4c26062	87) e5	b06b 1b	a)(o	cd025bb7	01136	35e	b0c04c4c8	d818d	237e	43d82a34d	ea991	ofac	bc429f4ca	d70f20	cd5)	526a2:	7bd)	636
feedback[127:0]	526a217bd22b6c416354b1fc61c48401	UU	4c26062	87 <mark>/</mark> e5	b06b 1b	a)(o	cd025bb7	0113	35e	b0c04c4c8	d818d	237e	43d82a34d	ea991	bfac	bc429f4ca	94070)af5	526a2	7bd)	636
▶ 🔣 round_key[127:0]	a60c63b6c80c3fe18925eec9a8f914d0	UU	3c4fcf09	88 05	766c2a	13)(7ff659737	3b887	a6d	00ad0bdb3	bc15f)11b	fd9300ca4	4fdca6	4eb	2f298d7f6	6e005	574	a60c6(b6c)	000
Interpretation in the second secon	6c	UU	01		02		04) 0	8	10) 2	D	40	X 8	0	∕ 1b	Х 3	i)	6		01
Vo sel	0																				
		¥1:1	10, 175 pe																		

Figure 16. Test Bench wave form of AES_ENC

In the above test bench the encryption of plaintext is x"2a179373117e3de9969f402ee2bec16b", under the key is x"3c4fcf098815f7aba6d2ae2816157e2b", which has to produce the cipher text as x"97ef6624f3ca9ea860367a0db47bd73a".

																		110.20)7 ns
Name Value	0 ns			20 ns			40 ns			60 ns			80 ns			100 ns	1		
le dk 0																			
Te rst 0																			
kiphertext[127:0] 2e2b34ca59fa4c883b2c8aefd44be966							320b	6a19978511d	cfb09dc0)21d84	2539							2e2b3	4ca5
#dec_key(127:0] 8e188f6fcf51e92311e2923ecb5befb4							a60	c63b6c80c3fe	18925ee	:9a8f9	14d0							8e 188	f6fcf
l done 1																			
plaintext[127:0] 340737e0a29831318d305a88a8f64332	UU)	940709af5	bc429	f4ca	ea991bfac	43d82	a34d	d818d237e	b0c04	:4c8	0113b35e6	dcd02	2 5667	e5b06b1ba	4c2606	287	340732	e0a	a03
clk_period 10000 ps									1000	0 ps									
mux_output[127:0] 2e2b34ca59fa4c883b2c8aefd44be966	320b6a	199) <mark>d24</mark>	c5b09	685a	0c048c7a6	258bd	2fda	0cd5b55d3	01beb	855	e7585fd63	9a46	3268d	49506a024	0848f8	8e92	(5cc)	2e2b3	4ca5
reg_output[127:0] 0848f8e92a8dc69a2be2f4a0bee33d19	(UU)	320b6a 199	d242c	31be	c5b09685a	0c048	:7a6	258bd2fda	Ocd5b	i5d3	0 1beb8855	e758	5 fd 63	9a463268d	49506	a024	0848f8	e92	2e2
Inv_mixcol_input[127:0] 340737e0a29831318d305a88a8f64332	(UU)	940709af5	bc429	f4ca	ea991bfac	43d82	a34d	d818d237e	b0c04	:4c8	0113b35e6	dcd02	2 5 bb7	e5b06b1ba	4c2606	287	340737	e0a	a03
inv_mixcol_output[127:0] 0e55de61148ff352ccbdc5db4a3bffa1	(UU)	262ad 1e8f	95c3e	c97d	4d8683dac	d3a9a	o54f	2329634cd	<mark>53fb4</mark>	97b	075e50f6d	b511	7845d	1a96de77f	e5982	71ef	0e55de	611	4fe
invsr_input[127:0] 0e55de61148ff352ccbdc5db4a3bffa1	(UU)	940709af5	95c3e	c97d	4d8683dac	d3a9a	o54f	2329634cd	53fb4	97b	075e50f6d	b511	7845d	1a96de77f	e5982	71ef	0e55de	611	a03
invsb_input[127:0] 4abdf3610e3bc5521455ffdbcc8fdea1	(UU)(b52c2eaf9	a6e79	0979	fef264da4	3f3db	i54d	fe03d54c2	7cae6	975	946acff60	b85a	2845b	3b5302771	30524	11ee	4abdf3	610	(1fc
feedback[127:0] 5ccd7ed8d74907489bed7d9f27739cf1	(UU)	d242c31be	c5b09	685a	0c048c7a6	258bd	2fda	0cd5b55d3	01beb	855	e7585fd63	9a46.	3268d	49506a024	0848f8	8e92	Sccd7e	d8d	cbe
Tound_key[127:0] 3c4fcf098815f7aba6d2ae2816157e2b	(UU)	a60c63b6c	6e005	:574	2f298d7f6	×4fdca6	4eb)	fd9300ca4	bc15f9	11b	00ad0bdb3	3b88	7a6d4	7ff659737	05766	2a3	3c4fcf0	988	8e1
▶ 號 round_const[7:0] 00		36) 1	þ	80	χ 4)	20) 1)	08	X	04	02	X 0	1	(0		36
1 is_first_round 0																			
	X1: 110).207 ns																	

Figure 17. Test Bench wave form of AES_DEC

In the above test bench the decryption of cipher text 32e2b34c959f94c883b2c8aefd44be9ee inputted is obtained as 8e188f6fcf51e92311e2923ecb5befb4.

Page 117



5. Conclusion

The 128-bit AES encoder, decoder are successfully implemented and tested for performance comparison. Thus the hardware architectures are much more dependent on the varying bit size than software implementation. This effect is enhanced with the employment of several cores in FPGA with smaller bit sizes. Although an additional core on FPGA might slightly decrease the maximum clock speed, it is overcome by the additional computational power provided by this extra point processor. The no of I/O, Slices in the proposed work are 386, 229 respectively with Spartan6 fpga with a minimum period of 5.813ns,maximum Frequency of 172.031MHz, minimum input arrival time before clock of 4.823ns,maximum output required time after clock of 5.588ns, throughput of 2.2Gbps. The no of I/O, Slices are 387, 264 respectively with Artix7 fpga with a minimum period of 3.397ns, maximum Grequency of 294.366MHz, minimum input arrival time before clock of 1.649ns, maximum output required time after clock of 1.669ns, throughput of 3.77Gbps. The algorithm can be implemented securely and efficiently in a wide variety of platforms and applications (e.g., 8-bit processors, ATM networks, voice & satellite communications, HDTV, B-ISDN, etc.). The algorithm can be implemented as a stream cipher, message authentication code (MAC) generator, pseudorandom number generator, hashing algorithm, etc.

References

1. Suresh, A., & Ajai, A. R. (2016). VLSI implementation of text to image encryption algorithm based on private key encryption. In 2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT) (pp. 4879–4881). IEEE. <u>https://doi.org/10.1109/ICEEOT.2016.7755647</u>.

2. Mishra, Z., Ramu, G., & Acharya, B. (2019). Hight speed low area VLSI architecture for LEA encryption algorithm. In Proceedings of the third international conference on microelectronics, computing and communication systems (pp. 155–160). Springer

3. Swetha, P. B., Sonti, V. K., & Murali, A. (2017). VLSI design for efficient RSD-Based ECC processor using Karatsuba algorithm. International Journal of Engineering & Technology, 7(1.5), 164–169. https://doi.org/10.14419/ijet.v7i1.5.9140.

4. Pandey, J., Gurawa, A., Nehra, H., & Karmakar, A. (2016). An efficient VLSI architecture for data encryption standard and its FPGA implementation. In 2016 International Conference on VLSI Systems, Architectures, Technology and Applications (VLSI-SATA) (pp. 1–5). IEEE. https://doi.org/10.1109/VLSI-SATA.2016.7593054

5. Hameed, M. E., Ibrahim, M. M., & Abd Manap, N. (2018). Review on improvement of advanced encryption standard (AES) algorithm based on time execution, differential cryptanalysis and level of security. Journal of Telecommunication, Electronic and Computer Engineering (JTEC), 10(1), 139–145.

6. Landge, I. A. G., & Mishra, B. (2016). Iterative architecture AES for secure VLSI based system design. In 2016 Symposium on Colossal Data Analysis and Networking (CDAN) (pp. 1–4). IEEE. https://doi.org/10.1109/CDAN.2016.7570938

7.Abdullah, A. M. (2017). Advanced encryption standard (AES) algorithm to encrypt and decrypt data. Cryptography and Network Security, 16

8.Siddesh, G., & Shruthi, J. (2017). AN EFFICIENT VLSI ARCHITECTURE FOR AES AND its FPGA IMPLEMENTATION. International Research Journal of Engineering and Technology, 4(6).
9. Keshava Kumar, "A Design Implementation and Comparative Analysis of Advanced Encryption Standard" 2020 8th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO) Amity University, Noida, India. June 4-5, 2020.

 K. P. Singh, and S. Dod. "An Efficient Hardware Design and Implementation of Advanced Encryption Standard (AES) Algorithm." IACR Cryptology ePrint Archive 2016 (2016): 789.
 U. Farooq r, and M. F. Aslam. "Comparative analysis of different AES implementation



techniques for efficient resource usage and better performance of an FPGA." Journal of King Saud University, Computer and Information Sciences 29, no. 3 (2017): 295-302.