Area and speed optimized Advanced Encryption Standard (AES) implementation on Xilinx-ISE

[1]Archana Senpuriya, [2]Prof. Divyanshu Rao

[1]M.Tech. Scholar, [2]Assistant Professor
Department of Electronics and Communication, SRIT, Jabalpur, MP, India

***Abstract*: Information security has become a very critical aspect of modern computing systems. With the global acceptance of internet, virtually every computer in the world today is connected to every other. While this has created tremendous productivity and unprecedented opportunities in the world we live in, it has also created new risks for the users of these computers. The users, businesses and organizations worldwide have to live with a constant threat from hackers and attackers, who use a variety of techniques and tools in order to break into computer systems, steal information, change data and cause havoc. The paper work aims at designing and implementing a secure data communication between any two users based on the realization of advanced Symmetric-key Cryptographic algorithm called Advanced Encryption Standard (AES) on an FPGA based processor.**

***Keywords*: AES, FPGA, VHDL, Semicustom, Full-Custom, RTL entry**

I. INTRODUCTION

Figure 1 shows AES Cryptographic technology is an important way to ensure information security, and is the key to information safety. Among all kinds of cryptographic algorithms, Advanced Encryption Standard Algorithm (AES) is preferred as it offers high security, efficiency, convenient usage, flexibility, and comprehensive performance. The AES algorithm is a symmetric block cipher that can encrypt, (encipher), and decrypt, (decipher), information. Encryption converts data to an unintelligible form called cipher-text. Decryption of the cipher-text converts the data back into its original form, which is called plaintext. The AES algorithm is capable of using cryptographic keys of 128, 192 and 256 bits to encrypt and decrypt data in the blocks of bits. AES cipher is specified as a number of repetitions of transformation rounds that convert the input plaintext into the final output of cipher text.

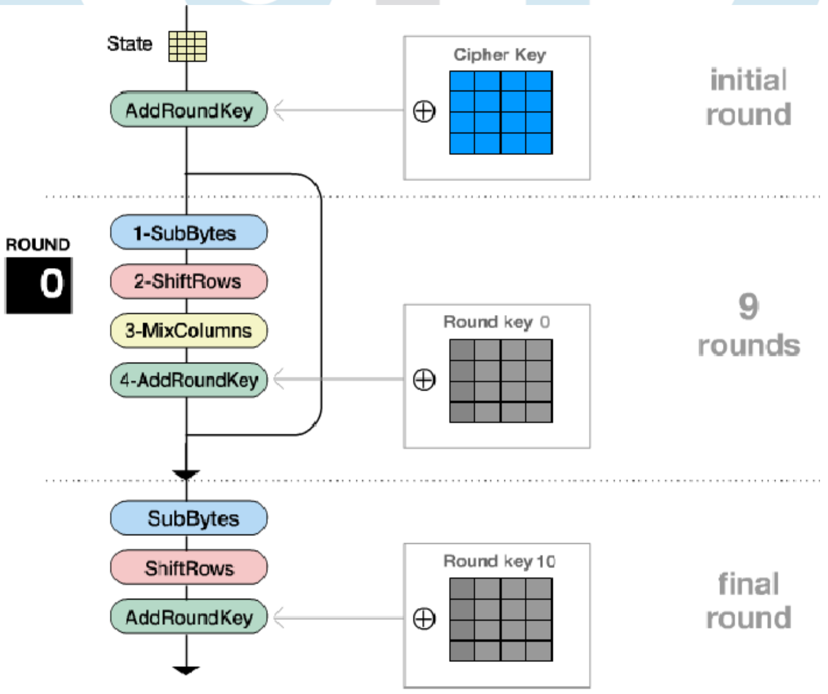


Fig1 Encryption process Block Diagram: AES

.

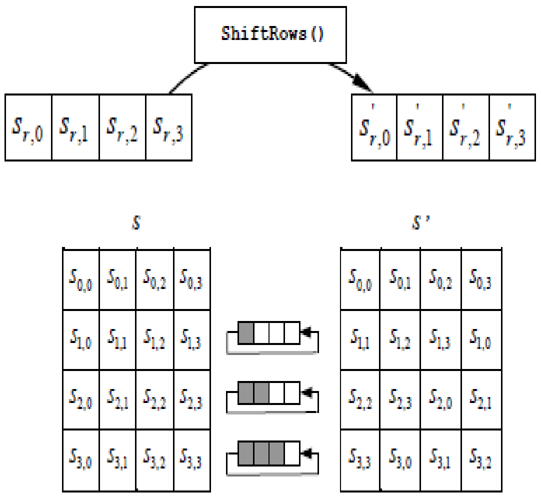
Each round consists of several processing steps, including one that depends on the encryption key. A set of reverse rounds are applied to AES is based on a design principle known as a Substitution permutation network. Unlike its predecessor, DES, AES does not use a Feistel network. AES operates on a 4 x 4 array of bytes called state in a matrix form. The algorithm consists of performing four different simple operation. These operations are: Sub Bytes, Shift Rows, Mix Columns and Add Round Key. AES operates on a 4x4 array of bytes (referred to as “state”). The algorithm consists of performing 4 different operations.

1.1 Sub-Bytes transformation: is a non-linear byte substitution that operates independently on each byte of the State using a substitution table (S-box).

Table 1 S-Box: Substitution

		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
	1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
	2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
	3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
	4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
	5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
	6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
	7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
	8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
	9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
	a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
	b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
	c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
	d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
	e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
	f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

1.2 Shift-Rows Transformation: The first row, r = 0, is not shifted. The shift value shift (r, Nb) depends on the row number, r, as follows (recall that Nb = 4):
shift(1,4) =1; shift(2,4) = 2 ; shift(3,4) = 3


ShiftRows()
$s_{r,0}$ $s_{r,1}$ $s_{r,2}$ $s_{r,3}$
$s'_{r,0}$ $s'_{r,1}$ $s'_{r,2}$ $s'_{r,3}$
s
s'
Figure 2 ShiftRows( ) cyclically shifts the last three rows in the state

1.3 MixColumns Transformation: The MixColumns() transformation operates on the State column-by-column, treating each column as a four-term polynomial. In the MixColumns step, each column of the state is multiplied with a fixed polynomial a(x).
In the MixColumns step, the four bytes of each column of the state are combined using an invertible linear transformation. The MixColumns function takes four bytes as input and outputs four bytes, where each input byte affects all four output bytes. Columns are considered with fixed polynomial a(x), given by

$$a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\} .$$

Let

$$s'(x) = a(x) \otimes s(x) :$$

$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix}$$

for $0 \le c < Nb$ MixColumns( ) operates on the state column-by-column
1.4 Add Round-Key Transformation: a Round Key is added to the output of MixColumn operation (state) by a simple bitwise XOR operation. For each round of operation, separate key is generated using Key Expansion.

1.5 Key Generation: Round keys are derived from the cipher key using Rijndael's key schedule. The AES algorithm takes the Cipher Key, K, and performs a Key Expansion routine to generate a key schedule. The Key Expansion generates a total of Nb (Nr + 1) words. The expansion of the input key into the key schedule proceeds as per the functions Rotword(), Subword(), Rcon[i/Nk], Xor operations.

II. METHODOLOGY

As discuss above AES requires lots of computation in four modules (Substitution block, shift rows, mix columns &Add round Key) of nine Full and one Sub-round. But it can be easily observe that Optimization in Area and speed possible only with block Key-Generator and S-Box only. Thesis works on new optimized S-box though Key-Generator technique remains unchanged.

Table 2 below shows the input and output of AES S8-Box, and the clear and complete observation gives the clue for the optimization in AES.

Substitution Box S-Box-8	
INPUT	OUTPUT
0000_0000	1111_1111
0000_0001	0111_1100
0000_0010	0111_0111
0000_0010	0111_1011
0000_0010	1111_0010
. . . .	. . . .
1111_1110	0111_0001
1111_1111	1111_0000

Table 2 shows the relation between input and output for s8 box (f8).Observation from table was that as for small size S-box (2-5 bit), memory based S-box is better area optimized and for bigger S box(more than 5 bit) Combinational architecture is better area optimized. Proposed work is a combination of memory and combinational architecture. The table show is relation between input and output for 8 bit S-box, thesis proposed architecture divided the total range 0-255 into 16 sub-ranges (0-15,16-31,32-47,48-63,64-79,80-97,96-111,112-127,128-143,144-159,160-175,176-191,192-207,208-223,224-239,240-255) isolation shown by orange lines. For each sub-range, upper four MSB of output (separated by Red line) are generated using 4 input K-map and lower four LSB of output are generated using Memory architecture. Figure 3 shows the architecture of proposed work which reflects the idea behind the new logic for architecture as explain above.

As the proposed work uses the S8-box explained above, and as proposed s8box is area and time efficient and as very much known S8-box use in AES one round about six time (i.e. one time in substitution, four times in Mix coulombs, one time in round key generation), and total nine full rounds requires 9x6 = 36 time use of s8-box and 2 times in sub-round which makes total 38 times use of s8-box in single plaintext to cipher-text generation. So If proposed S8-box is optimized in terms of area and speed the Full AES is also optimized as S8-box gets in use 38 times.

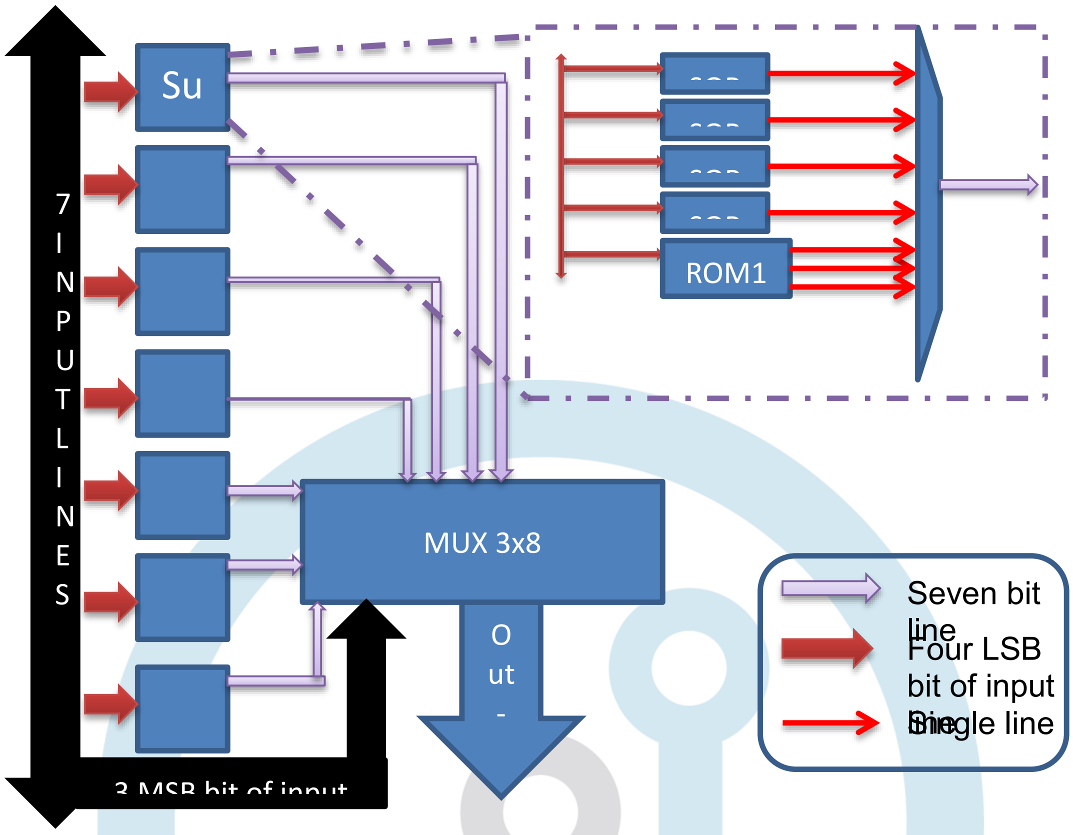


Figure 3: Proposed architecture S8 box

III. RESULTS

3.1 Tool and language used

Tool: Xilinx ISE- It is a software tool produced by Xilinx for synthesis and analysis of HDL designs.

Language used: Verilog HDL: Verilog, standardized as IEEE 1364, is a hardware description language (HDL) used to model electronic systems. It is most commonly used in the design and verification of digital circuits at the register-transfer level of abstraction

Platform Used: family- Vertex4, Device-XC4VLX80, Package-FF1148. Target FPGA is a Vertex FGPA because the same platform is been used by base papers.

3.2 Simulation and synthesize of Proposed work

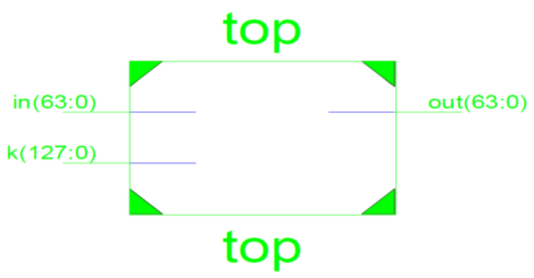
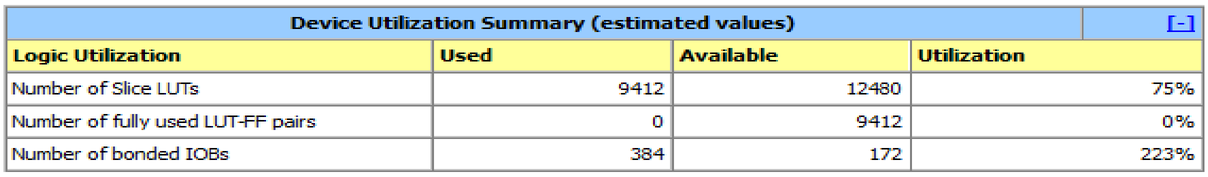


Figure 4: Top RTL of proposed work: AES Encryption



Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice LUTs	9412	12480	75%
Number of fully used LUT-FF pairs	0	9412	0%
Number of bonded IOBs	384	172	223%

Figure 5: Synthesis Summary of proposed work: AES Encryption

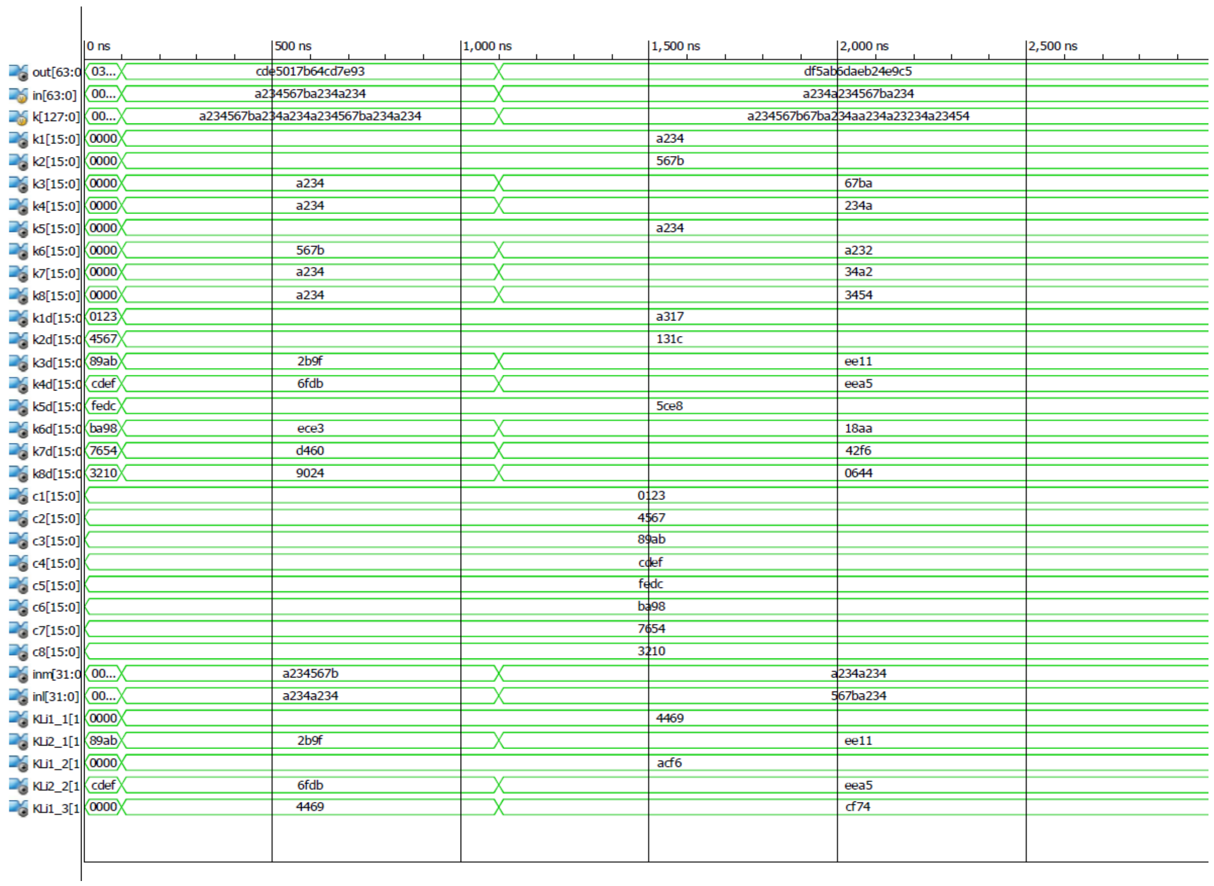


Figure 6: simulation and RTL schematic of proposed work

Results: From the simulation as shown in above slides
Key:A234567ba234a234a234567ba234a234
Result:-1
Output: Cde5017b64cd7e93
Input: A234567ba234a234
Output^Input: 6fd15700c6f9dca7 Avalanche: 41 bit change/64 bit
Result:-2
Output: Df5ab6daed24e9c5
Input: A234a234567ba234
Output^Input: 7d6e14eebb5f4bf1
Avalanche: 45 bit change/64 bit

Table 3: Simulation outcomes for each module

Parameters	Design of FI	Design of FO	Design of FL	Design of Sbox-7	Design of Sbox-9	Complete KASUMI module
No. of slice	429	1379	18	26	157	8401
No. of LUT's	782	2541	32	52	289	15468
No. of IOB's Logical Time delay	13.04 ns	11.216 ns	4.303 ns	6.067 ns	7.279 ns	33.64 ns

Table 4 : comparative results

	Parameters	**Base [1]**		**Base[2]**		**Proposed work**	
S-box design		S-box 7 (S7)	S-box 9 (S9)	S-box 7 (S7)	S-box 9 (S9)	S-box 7 (S7)	S-box 9 (S9)
	No. of slice	34	169	-	-	26	157
	Logical Time delay (ns)	-	-	-	-	6.067	7.279
Overall Kasumi encryption design	No. of slice	8784		8770		8401	
	Logical Time delay (ns)	34.01		-		33.64	

IV CONCLUSION

The work is implemented of FPGA which makes proposed work a semicustom design as known semicustom design always lack behinds compare to full-custom design in term of Area, speed and power. In future proposed work can be implemented at transistor level (i.e. Full-custom). The large number of potential subscribers and the high end services to provide may have great challenges in terms of guaranteed confidentiality and integrity of both information and signalling. An optimised and compact hardware design of the AES algorithm has been described in this thesis work, as well as with the results of its implementation in FPGA technology. These proposed S8-box method might be use to design high performance compact implementations of Feistel-like block ciphers (AES, IDEA etc.). Not only does this proposal achieve a high performance, but is one of the most cost efficient designs in terms of area.

It can be concluded as discuss that S8-box is an important requirement in AES cipher generation and it get use 38 times for generating 64 bit cipher-text from plaintext. proposed S8-box 7.741 ns time delay and only 64 slices, which is less as compare to all existing works which are been discuss in chapter 3. The comparative results in chapter 6 also shows that proposed work is optimized in terms of Area and speed as compare to existing work.

References

[1] K. Kumar, K. R. Ramkumar and A. Kaur, "A Design Implementation and Comparative Analysis of Advanced Encryption Standard (AES) Algorithm on FPGA," 2020 8th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), 2020, pp. 182-185, doi: 10.1109/ICRITO48877.2020.9198033.

[2] S. Chen, W. Hu and Z. Li, "High Performance Data Encryption with AES Implementation on FPGA," 2019 IEEE 5th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing, (HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS), 2019, pp. 149-153, doi: 10.1109/BigDataSecurity-HPSC-IDS.2019.00036.

[3] A. Sideris, T. Sanida and M. Dasygenis, "Hardware Acceleration of the AES Algorithm using Nios-II Processor," 2019 Panhellenic Conference on Electronics & Telecommunications (PACET), 2019, pp. 1-5, doi: 10.1109/PACET48583.2019.8956285.

[4] R. V. Kshirsagar and M. V. Vyawahare, "FPGA Implementation of High Speed VLSI Architectures for AES Algorithm," 2012 Fifth International Conference on Emerging Trends in Engineering and Technology, 2012, pp. 239-242, doi: 10.1109/ICETET.2012.53.

[5]. Tomas balderas-contreras, rene cumplido , claudia feregrino-uribe, “On the design and implementation of a RISC processor extension for the KASUMI encryption algorithm”, T. Balderas-contreras et al. / Computers and electrical engineering 34 (2008) 531–546

[6]. Sima I., Tarmurean D., Greu V, Diaconu A.‘XXTEA, an alternative replacement of KASUMI cipher algorithm in A5/3 GSM and f8, f9 UMTS data security functions’ 9th International Conference on Communications (COMM), volume 1, pp 328-333

[7]. Ren fung, ying-jian, Fu Xiao-bing, ‘A Small and Efficient Implementation of KASUMI’, IEEE Explore, WASE International Conference on Information Engineering, volume 2, pp 377-380, 10-11 july, 2011

[8]. Hui Shi Yuanqing Deng Yu Guan Peng Jia Fengli Ma, Analysis of the Avalanche Property of the KASUMI Algorithm, Automatic Control and Artificial Intelligence (ACAI 2012), International Conference on, IEEE Explore, 3-5 March 2012

[9]. P. Kitsos, M. D. Galanis, and O. Koufopavlou,“High-speed hardware implementations of the kasumi block cipher” ISCAS 2004, ©2004 IEEE.