International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395 -0056
Volume: 03 Issue: 07 | July-2016 www.irjet.net p-ISSN: 2395-0072

Design and ASIC Implementation of Modified Rijndael Cipher

Sushikshith H. Gowda[1], Dr. Aravind H. S.[2], Usha S. M.[3]

[1] *M.Tech., VLSI Design & Embedded Systems, Dept. of E&C Engineering, JSSATE, Bengaluru, India*
[2]*Professor and Head, Dept. of E&C Engineering, JSSATE, Bengaluru, India*
[3]*Assistant Professor, Dept. of E&C Engineering, JSSATE, Bengaluru, India*

Abstract - *In this paper, Rijndael with modified Sbox and ShiftRow transformations is designed in system verilog and its ASIC implementation is performed. The objective of the project is to increase the complexity and strength of the cipher against external attacks by implementing the proposed modifications. The inputs to the proposed Rijndael are 128 bits in length. The encryption and decryption modules are verified for functionality on Cadence SimVision and is synthesized using Cadence RTL Compiler. Physical design flow is followed to perform ASIC Implementation of the design on the synthesized netlist using Cadence SoC Encounter in 45nm technology.*

Key Words: **Modified Rijndael, Triple substituted S-box, Modified ShiftRow with mode operation**

1. INTRODUCTION

Rijndael symmetric block cipher is the advanced encryption standard, because it is the safest cipher offering high modularity, non-linearity and diffusion characteristics without the Feistel structure. Although the actual Rijndael [1] is not practically breakable, the innovations in technology and processing speeds can make the theoretical attack possible. Hence, in this paper, an attempt to increase the non-linearity and diffusion attributes have been made by proposing a modified Rijndael cipher with two simple design modifications that would not only increase the complexity, but also generates two different cipher texts for a plain text and cipher key thereby doubling the effort of the attacker to break the encryption. Rest of the paper is organised as follows: Section 2 covers the literature survey, section 3 explains the modified Rijndael cipher, section 4 addresses the implementation, tests and results of the modified Rijndael design, and finally the conclusion is covered in section 5.

2. LITERATURE SURVEY

In Rijndael's case, modifications presented span from one to every transformation of the algorithm. Some proposals concentrate on reducing the hardware resources used and improving the performance and throughput whereas some other proposals desire to add complexity to the existing cipher. Some of the modified proposals have been reviewed in this work.

Ahmed A. Mohamed in [2] proposes an additional step such as Mirror & Inverse Mirror in the round transformations of encryption & decryption respectively. Ahmed has implemented his design using VHDL and Xilinx on a virtex FPGA. The modified version of Rijndael proposed by Marian Cretu [3] analyses the change in accessing order of Sbox as compared to the original algorithm. A personalized version of the Rijndael by Ion-SIMA [4] analyses cipher text correlations and ASCII characters frequencies by modifying ShiftRow & MixColumn transformations pseudo-randomly. A new design for Key schedule expansion is proposed by Salasiah Sulaiman in [5] includes an additional ShiftColumn step in the conventional Rijndael key expansion stage.

A Design that proposes 2D Rotations in AES for Protection of Smart Cards proposed by Umadatta A. can be seen in [6]. The two modified designs proposed by Sliman Arrag in [7] are based on converting the static Rijndael Sbox into a dynamic S-box that is dependent on the Master key. K. Rahimunnisa's proposed work for Modified S-Box in [8] is designed by composing 2 transformations and consumes lesser power compared to the normal Sbox. Faiz Yousif Mohammad [9] proposes a S-box derived from variable mapping is based on generating a parameter using the key and the derived sub keys that's used to randomly shift the S-box's substitution to a different location.

A multiple S-box based dual key transformation proposed by Nada Hussein M. Ali [10] where the first key is used to generate multiple keys which have their inverses and eight different keys are used to obtain eight different S-box so that 16 different keys are possible using the eight S-box for the second key. An S-box designed by Bodhisatwa Mazumdar that was developed with DPA resistance in mind is RAIN S-Box presented in [11]. A dual key technique proposed by Mohammed M. Alani in [12] makes use of 2 keys where the second key is used only for the final round. An implementation of AES-128 on FPGA proposed by Hrushikesh S. Deshpande in [13] swaps the ShiftRow and ByteSub transformations in every round. Bhoopal Rao Gangadari's work in [14] indicates that the robustness of the polynomial chosen for S-box design by the Rijndael creators is the highest compared to several other possible polynomials for cryptographic properties like SAC, Entropy, Non-linearity & CIB.

All the above modified proposals along with few other proposals have been reviewed in [15].

3. MODIFIED RIJNDAEL

The proposed modified Rijndael has the advantage of performing individual encryption and decryption of different plain and cipher texts with different keys respectively. This is clearly visible from the top level block diagram shown in the Fig - 3.1.

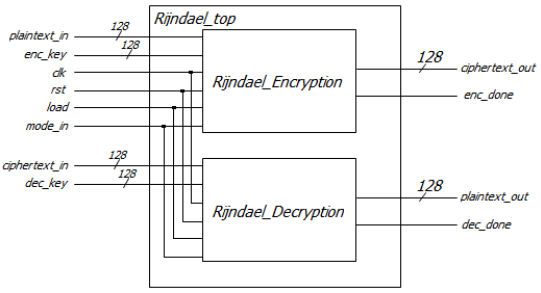

Fig - 3.1: Top level Block diagram of the implemented design

Fig - 3.2 shows the transformations in encryption and decryption processes. The encryption process starts by adding the initial key to the plain text and the obtained sum is the input to the round transformations performed. The number of rounds are identical to Rijndael with ten rounds for 128 bit input data which are arranged as a 4x4 state matrix.

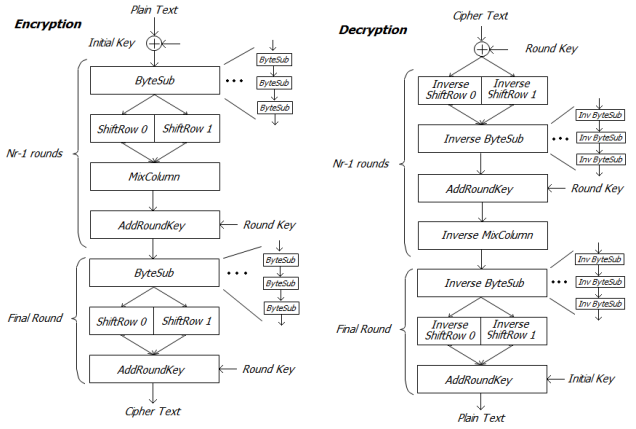

Fig - 3.2: Block Diagram of Proposed Modified Rijndael Encryption and Decryption

The nine rounds are identical with ByteSub, ShiftRow, MixColumn and AddRoundKey transformations. The final round omits the MixColumn transformation. These transformations are detailed below.

3.1 Modified ByteSub: This transformation substitutes the plain data byte by a corresponding random byte from the modified S-box as shown on Fig - 3.3. The design of modified S-box used in this work is a triple substitution S-box derived from the Rijndael S-box by a simple substitution process. This modification is derived on paper and implemented as one piece in the design rather than performing calculations in real-time during the operation of the design to avoid additional computation resulting in increased power dissipation. The modified S-box that was obtained by double substitution is discarded due to its vulnerability.

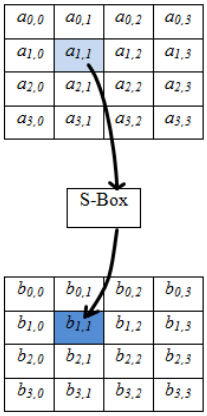

Fig - 3.3: Byte Substitution using Sbox

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0F	CA	E6	FD	A7	D2	C2	24	F2	10	97	A1	EA	AB	AA	07
1	92	7D	C1	16	D8	1F	E0	64	2A	52	80	B6	1D	3B	09	F4
2	D3	20	86	68	6B	9D	45	B3	AD	6F	35	32	0A	EF	C6	CB
3	89	B4	F7	31	95	60	7F	6C	A6	DD	BD	46	1E	4B	9A	5E
4	7C	CE	A3	3A	79	DB	AE	E1	63	98	42	3C	06	82	59	CF
5	55	B2	FB	FC	A9	E7	E8	12	77	C0	E4	C9	F6	A5	02	7E
6	51	9E	91	76	A2	11	2E	88	9F	EE	F5	B5	ED	E9	B9	25
7	3E	67	01	8F	84	58	C5	8E	4D	2F	5B	54	74	47	D7	D5
8	7A	BB	FF	8B	8A	C4	AF	8C	9C	4A	0D	CC	1A	29	48	73
9	70	FE	5F	44	DC	D9	D0	1C	BE	34	50	2D	A4	6A	F1	56
A	F8	26	CD	85	E2	A8	05	D6	3F	33	81	AC	0C	E5	F9	4E
B	22	9B	B8	EB	4C	7B	15	66	53	C8	08	17	E3	57	69	04
C	BF	65	75	C7	DE	36	5D	8D	14	78	4F	BA	6D	DA	27	F3
D	D1	37	03	C3	00	21	2C	62	DF	90	39	B1	1B	BC	49	2B
E	41	83	5A	13	99	96	D4	93	FA	40	F0	72	3D	B0	18	0B
F	43	23	5C	0E	30	19	71	6E	EC	28	61	38	94	B7	87	A0

Fig - 3.4: Triple substitution Sbox, 3S_Sbox

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	D4	72	5E	D2	BF	A6	4C	0F	BA	1E	2C	EF	AC	8A	F3	00
1	09	65	57	E3	C8	B6	13	BB	EE	F5	8C	DC	97	1C	3C	15
2	21	D5	B0	F1	07	6F	A1	CE	F9	8D	18	DF	D6	96	66	79
3	F4	33	2B	A9	99	2A	C5	D1	FB	DA	43	1D	4B	EC	70	A8
4	E9	E0	4A	F0	93	26	3B	7D	8E	DE	89	3D	B4	78	AF	CA
5	9A	60	19	B8	7B	50	9F	BD	75	4E	E2	7A	F2	C6	3F	92
6	35	FA	D7	48	17	C1	B7	71	23	BE	9D	24	37	CC	F7	29
7	90	F6	EB	8F	7C	C2	63	58	C9	44	80	B5	40	11	5F	36
8	1A	AA	4D	E1	74	A3	22	FE	67	30	84	83	87	C7	77	73
9	D9	62	10	E7	FC	34	E5	0A	49	E4	3E	B1	88	25	61	68
A	FF	0B	64	42	9C	5D	38	04	A5	54	0E	0D	AB	28	46	86
B	ED	DB	51	27	31	6B	1B	FD	B2	6E	CB	81	DD	3A	98	C0
C	59	12	06	D3	85	76	2E	C3	B9	5B	01	2F	8B	A2	41	4F
D	96	D0	05	20	E6	7F	A7	7E	14	95	CD	45	94	39	C4	D8
E	16	47	A4	BC	5A	AD	02	55	56	6D	0C	B3	F8	6C	69	2D
F	EA	9E	08	CF	1F	6A	5C	32	A0	AE	E8	52	53	03	91	83

Fig - 3.5: Triple substitution inverse Sbox, 3S_iSbox

The triple substituted S-box seen in Fig - 3.4 provides 98.72% uniqueness in diffusion and only 1.28% of substitutions overlap with those of Rijndael S-box. Fig - 3.5 shows inverse Triple substitution S-box that's used in Inverse ByteSub of the Decryption process.

3.2 Modified ShiftRow: This transformation cyclically shifts every row of the state with an offset an offset difference greater than or equal to two. Depending on the mode input, two shifts LS1302 & LS2031 is performed. When mode_in=1, Left Cyclic Shift of rows 0,1,2,3 by an offset of 1,3,0,2 is performed respectively, indicated by LS1302 and when mode_in=0, LS2031 is performed as in Fig - 3.6.

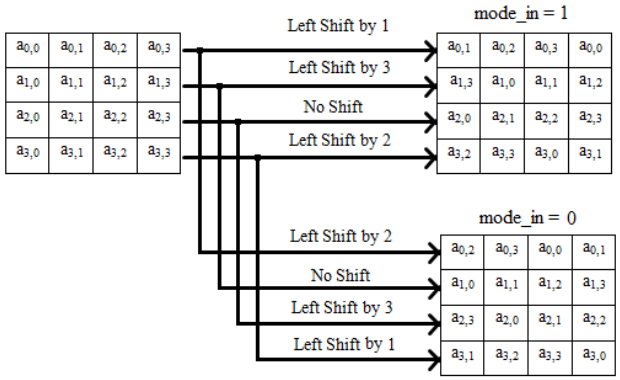

Fig - 3.6: Modifed ShiftRow with mode operation

For Modified Inverse ShiftRow, Right cyclic shift of the same offset is performed, i.e., RS1302 when mode_in=1 and RS2031 when mode_in=0 as shown in Fig - 3.7.

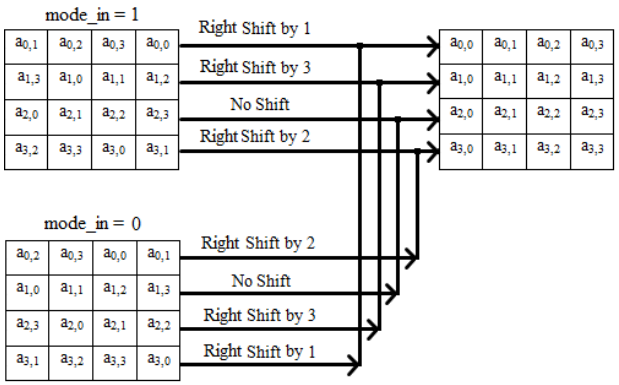

Fig - 3.7: Modified Inverse ShiftRow with mode oepration

3.3 MixColumn: In this transformation every column of the state is multiplied modulo (x^4+1) with a fixed polynomial called c(x), which is given by $c(x) = (03).x^3 + (01).x^2 + (01).x + (02)$.This is illustrated in Fig - 3.8 and Fig - 3.9.

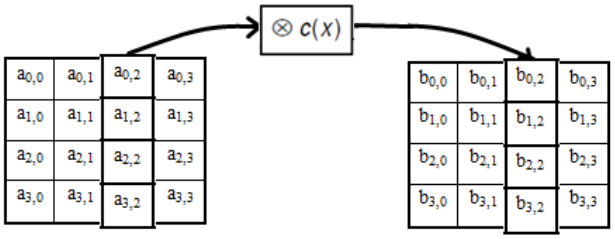

Fig - 3.8: MixColumn operation on state

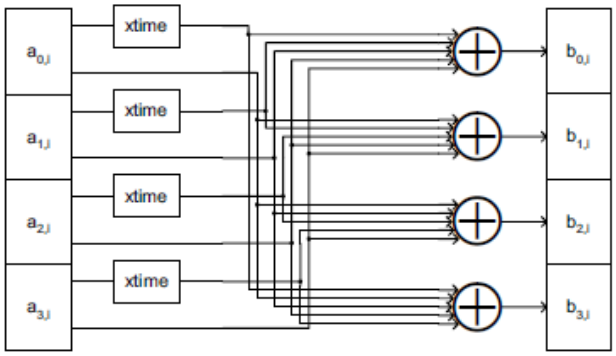

Fig - 3.9: MixColumn operation for a column of the state.

Inverse MixColumn as shown in Fig - 3.10, multiplies each column of the state by a specific polynomial d(x), defined by $((03).x^3 + (01).x^2 + (01).x + (02)) \otimes d(x) = '01'$. The polynomial d(x) is given by: $d(x) = (0B).x^3 + (0D).x^2 + (09).x + (0E)$

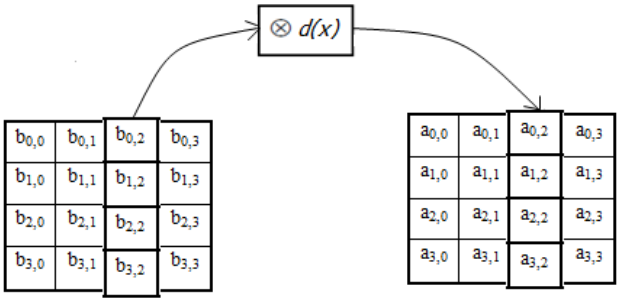

Fig - 3.10: Inverse MixColumn operation on the state

Fig - 3.11 shows the Inverse MixColumn operation for a column of the state. Fig - 3.12 shows Xtime operation which modulo multiplies a one byte number by 02.

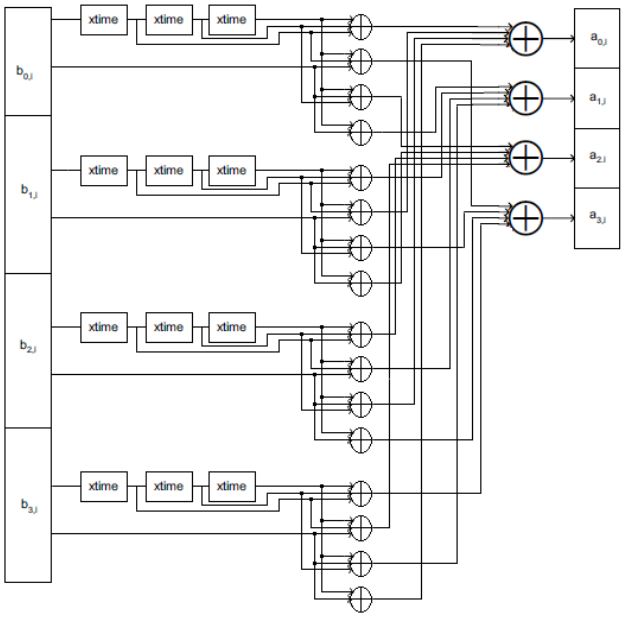

Fig - 3.11: Inverse MixColumn operation for a column of the state

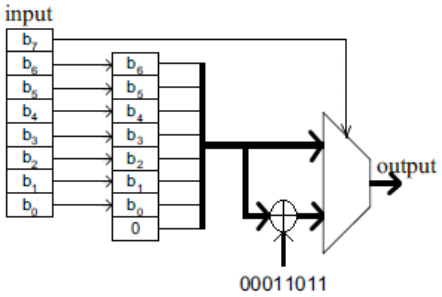

Fig - 3.12: Xtime Operation

3.4 Round Key Addition: The round key is added to the state via bitwise XOR operation. The round key for every round is derived from the user specified Cipher Key by means of key schedule. The key schedule expands the cipher key into 10 round keys of 128 bits each used in 10 rounds. Every round key generated is expanded from its previous round key.

For decryption, the order of the round key access is opposite to that of the encryption process.

4. IMPLEMENTATION, TESTS & RESULTS

4.1 Simulation of the Design

The system verilog description of modules rijndael_top, rijndael_enc_top, rijndael_dec_top, key_expand, rcon, 3S_Sbox and 3S_iSbox are compiled and elaborated. The design is then simulated on Cadence SimVision and the simulations are viewed on the waveform window and the functionality is verified for encryption and decryption. The simulations are as shown in Fig - 4.1 & Fig - 4.2 for encryption & decryption respectively.

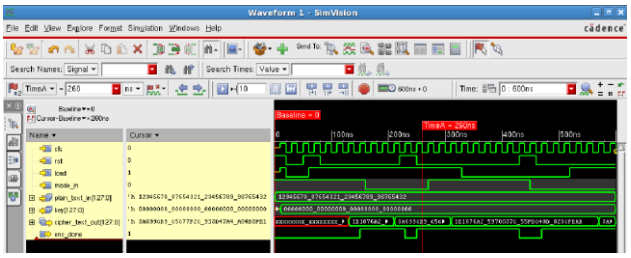

Fig - 4.1: Encryption simulation output

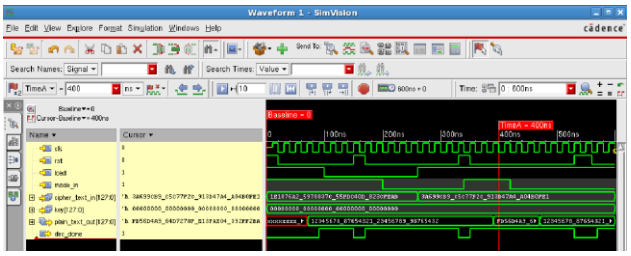

Fig - 4.2: Decryption simulation output

The above simulations analyse the various combinations of inputs with respect to the mode_in reset and load input signals. It is observed that the output is obtained after two clock cycles and the output done_enc/ done_dec signal goes high after the encryption/decryption is completed to indicate that the process is complete. The mode_in input remains constant for a session of encryption and decryption.

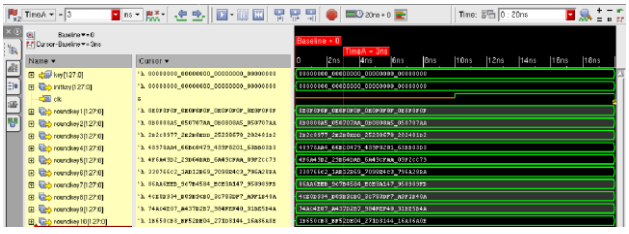

Fig - 4.3: Key Schedule output simulation

The key schedule output can be seen in Fig - 4.3. The round keys generated for a 128 bit 0's cipher key as seen in the simulation shows that key schedule also contributes for the complexity of the cipher.

4.2 Logic Synthesis of the Simulated Design

The constraints are applied to the simulated design and its logic synthesis is performed using a 45nm technology library on Cadence RTL Compiler Ultra. The timing slack and power dissipation for various synthesis frequencies are as shown in Table - 4.1 and Table - 4.2 respectively.

Modified Rijndael was synthesized at 100 MHz clock frequency with slightly relaxed constraints to achieve better timing against a power trade off.

The timing, power and area reported after synthesizing the design for 100 MHz clock frequency at medium effort is tabulated in Table - 4.3.

Table - 4.1: Comparison of Timing Slack for various synthesis frequencies

Operating Frequency(MHz)	Synthesis Effort	Generic slack(ps)	Mapped Slack (ps)	Timing Slack (ps)
250	Medium	-11044.1	-9085.5	-8491.1
200(CG:en)	High	-11794.1	-11106.7	-10331.4
175	High	-10189.1	-9469.2	-8076.5
100	High	-8044.1	-7354.2	-6576.1
50	Medium	-2950.8	-1125.5	-463.5

Table - 4.2: Comparison of Total power for various synthesis frequencies

Operating Frequency(MHz)	Cells	Leakage Power(nW)	Dynamic Power(nW)	Total Power(nW)
250	268023	43265.767	847821539.635	847864805.402
200	273709	45941.710	555935667.605	555981609.315
175	270111	43492.937	510481549.440	510525042.377
100	274467	43571.426	383634639.101	383678210.526
50	248287	36332.858	332727994.787	332764327.664

Table - 4.3: Reported Timing, total power and area for the final design

Operating Frequency(MHz)	Synthesis Effort	Generic slack(ps)	Mapped Slack (ps)	Timing Slack (ps)
100	Medium	-4134.6	-1907.8	-1297.2

Operating Frequency(MHz)	Cells	Leakage Power(nW)	Dynamic Power(nW)	Total Power(nW)
100	264920	38522.564	530764400.947	530802923.511

Operating Frequency(MHz)	Instance	cells	Cell area	Net area	Total area
100	Rijndael_top	264920	673802	294314	968116

The timing slack observed after synthesis is -1297.2ps. The total power dissipated equals 530mW and most of it is dynamic power occurring due to transiting nodes in the design.

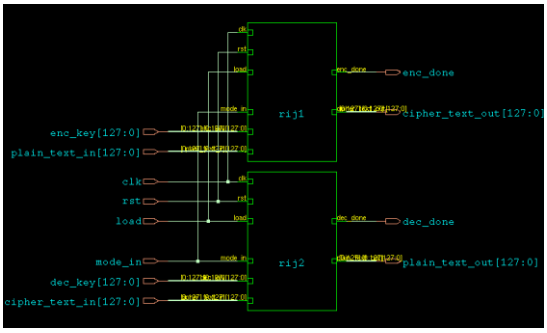

Fig – 4.4: Snapshot of schematic of the rijndael_top module.

The area consumed for synthesis is 968116um^2. The timing slack is reduced during placement and CTS optimizations performed during the ASIC implementation. Fig - 4.4 shows the snapshot of the schematic of the top module modified Rijndael where rij1 and rij2 are encryption & decryption instances respectively.

4.3 ASIC Implementation of the Synthesized Design

ASIC Design flow is followed to obtain physical layout of the design using Cadence SoC Encounter tool. The steps followed were Synthesis, Floor Planning, Power Planning, Placement, Clock Tree Synthesis and Routing.

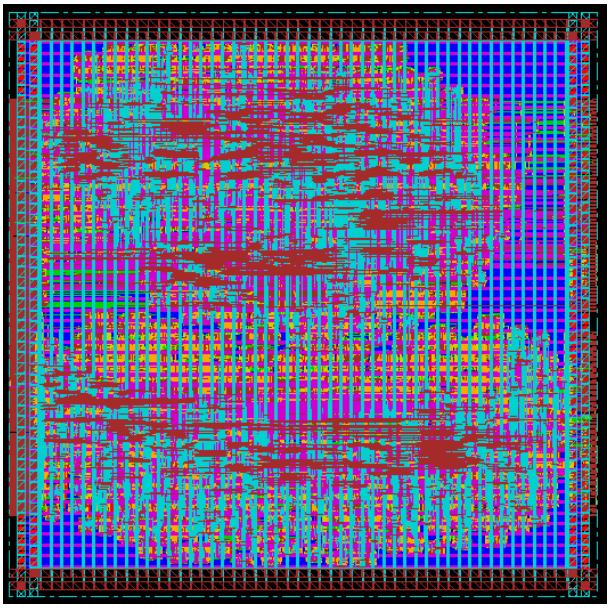

Fig – 4.5: Design after global routing

After synthesis, the gate level netlist is imported on Encounter. Standard cell rows, tap cells are added, pins placed, Power ring and stripes are drawn and connected to the standard cell power rails. The design is placed and verified for geometry. The clock tree is synthesized; final detailed routing is performed and optimized. Fig - 5.8 shows the Physical layout of the design after detailed global routing.

5. CONCLUSION

The Modified Rijndael cipher for encryption and decryption of 128 bit inputs is designed using system verilog. The modified ByteSub transformation is equivalent to performing the byte substitution three times using Rijndael Sbox thus increasing the non-linearity depth by 3 times for one round. For ten rounds, it is equivalent to increasing the non-linearity depth by 10 folds of one round's equivalent depth.

Similarly, diffusion attribute by ShiftRow transformation is increased by the changed offset difference between row shifts to two or more. With the ability to generate two cipher

texts according to the mode input, it adds more confusion and complexity to the attacker's effort. Thus, the proposed design is expected to offer higher complexity than the Rijndael itself.

The top module offers an advantage to perform encryption and decryption simultaneously for different data inputs. This gives the flexibility to communicate between two different parties simultaneously while sending data to one party and receiving data from another party.

The Modified Rijndael is simulated using Cadence SimVision, synthesized at 100MHz using Cadence RTL Compiler Ultra to yield a total power of 530mW and total area of 968116um2 with a timing slack of -1297.2 ps. The synthesized netlist is ASIC implemented and the timing is further optimized using Cadence SoC Encounter.

REFERENCES

[1] Joan Daemen and Vincent Rijmen, The Design of Rijndael, AES - The Advanced Encryption Standard, Springer-Verlag 2002 (238 pp.)

[2] Ahmed A. Mohamed, Ahmed H. Madian, "A Modified Rijndael Algorithm and its Implementation using FPGA", 17th IEEE International Conference on Electronics, Circuits and Systems, pp. 335-338, 2010.

[3] Marian Cretu, Cristian-Gabriel Apostol, " A Modified Version of Rijndael Algorithm Implemented to Analyse the Ciphertext Correlation for Switched S-Boxes", 9th International Conference on Communicatons (COMM), pp.21-23, June 2012.

[4] Ion SIMA, Adrian-Viorel DIACONU, Marian CRETU, "Analysis of Modified ShiftRows and MixColumns Transformations in Rijndael Algorithm", International Conference on Electronics, Computers and Artificial Intelligence, pp.1-4, June 2013

[5] Salasiah Sulaiman, Zaiton Muda, Julia Juremi, " A New Approach of Rijndael Key Schedule", International Conference on Cyber Security, Cyber Warfare & Digital Forensic(CyberSec),pp. 23-27, June 2012.

[6] Umadatta A., Dr. N. Chandra Sekhar Reddy, Arvind Kumar "Design of a AES 2D Rotations Algorithm used in Protection of Smart Cards", International Journal of Advanced Research in Computer Science and Software Engg.,Vol.4 No.5, May 2014.

[7] Sliman Arrag, Abdellatif Hamdoun, Abderrahim Tragha, Salah Eddine Khamlich, "Implementation of Stronger AES by using Dynamic S-Box Dependent of Master Key", Journal for Theoretical and Applied Information Technology, Vol . 53 No.2, July 2013, pp. 196-204.

[8] K. Rahimunnisa, K. Rajeshkumar, " A Novel Approach towards Implementation of AES and Performance Comparison with Modified S-box", International Journal of Micro and Nano Electronics, Circuits and Systems, 3(1),2011, pp. 29-33.

[9] Faiz Yousif Mohammad, Alaa Eldin Rohiem, Ashraf Diaa Elbayoumy, "A Novel S-box of AES Algorithm Using Variable Mapping Technique", 13th International Conference on Aerospace Science and Aviation Technology, ASAT-13, May 2009, pp.1-10.

[10] Nada Hussein M. Ali, Abdul Monem S. Rahma, Abdul Mohsen Jaber , "Encryption using Dual Key Transformation based on Creation of Multi S- Boxes in AES Algorithm", International Journal of Computer Applications, Vol. 83 No.10, Dec 2013.

[11] Bodhisatwa Mazumdar, Debdeep Mukhopadhyay and Indranil Sengupta, "Design for Security of Block Cipher S-Boxes to Resist Differential Power Attacks", 25th International Conference on VLSI Design, 2012

[12] Mohammed M. Alani, "Measuring the Effect of AES Encryption on VoWLAN QoS", International Conference on Software, Telecommunications & Computer Networks (SoftCON), pp. 51-54, 2010.

[13] Hrushikesh S. Deshpande, Kailash J. Karande, AltaafO. Mulani, "An efficient implementation of AES algorithm on FPGA", International Conference on Communication and Signal Processing, April 2014.

[14] Bhoopal Rao Gangadaril, Shaik Rafi Ahamed, "Analysis and Algebraic Construction of S-Box for AES algorithm using Irreducible Polynomials" 978-1-4673-7948-9/15/2015 IEEE

[15] Sushikshith H. Gowda, Dr. Aravind H.S., "A Review of Modified Rijndael Proposals", International Journal of Information Technology and Computer Engineering, e-ISSN: 2455-529.