International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2013): 4.438

Design and Implementation of Rijindael's Encryption and Decryption Algorithm using NIOS-II Processor

Monika U. Jaiswal[1], Nilesh A. Mohota[2]

[1] Student, Electronics Department, JDCOEM, Nagpur, India

[2]Professor, Electronics Department, JDCOEM, Nagpur, India

Abstract: ***One of the foremost vital problems in communication customary is that the secure transport protocols. This paper can offer a doable resolution for Rijindael's encryption and decoding algorithmic program using NIOS II processor, provided by ALTERA to be enforced in FPGA. We are going to see the performance of Rijindael's AES using NIOS II/e (economic), NIOS II/s (standard) and NIOS II/f (fast). The suggested system is capable of encrypting and decrypting 128,198 and 256 bits of data. The FPGA has the potential of data processing and hardware modification. The NIOS II is a versatile embedded processor family that represents high performance, lower overall cost, power consumption, complexity combining several functions into one chip. The look of the Rijindael algorithmic program supported "NIOS II + FPGA" are able to do a better processing speed whereas it occupies comparatively low resources. The inputs and the control of an AES algorithmic program is written in C language and is interfaced with the system using general purpose input and output (GPIO) and also the management part is enforced in software in NIOS II integrated development environment (IDE). The implementation is completed on Cyclone II FPGA kit. The results are analysed on the personnel computer (PC) in IDE console window.***

Keywords: Rijindael's algorithm, AES, DES, FPGA, SoPC, NIOS II

1. Introduction

FPGA (Field Programmable Gate Array) offers one or additional silicon chip implementations just like the NIOS II processor created by ALTERA just for FPGA. For an extended time, the data encryption standard (DES) was thought-about as a standard for the symmetric key encoding. DES includes a key length of 56 bits. However, this key length is currently thought of little and may simply be broken. For this reason, the National Institute of Standards and Technology (NIST) opened a proper necessitate algorithms in Sep 1997. A group of fifteen AES (Advanced Encryption Standard) candidate algorithms were declared in August 1998. In August 2000, NIST designated 5 algorithms: Mars, RC6, Rijndael, Serpent and Twofish as the final competitors. Finally the Rijindael algorithmic rule was winner and the new Advance Encryption Standard (AES) suggested by the US National Institute of Standards and Technology (NIST) [1]. It's a block cipher algorithmic rule that encrypts blocks of 128, 192 or 256 bits. Therefore, the matter of breaking the key becomes harder.

2. Advanced Encryption Standard (AES)

AES is an iterated block cipher with a fixed size of 128 and a variable key length. The input is given as a state which is a rectangular array of bytes and the block Size is of 128 bits, which is 16 bytes; the Rectangular array is of dimensions 4x4. The key is similarly pictured as a rectangular array with four rows. The number of columns of the key, denoted Nk, is equal to the key length divided by 32. AES uses a variable number of rounds, which are fixed for a particular AES type as mentioned below.

Table 1: Types of AES and its parameters

AES Type	Key Length (N_k Words)	Block Size (N_b Words)	Number of Rounds (N_r)
AES -128	4	4	10
AES – 192	6	4	12
AES – 256	8	4	14

The different stages involved in Rijindael's AES algorithm, i.e. in AES encryption and decryption are as mentioned below.

2.1 Encryption Process

Encryption converts data to an unreadable form which is called as cipher-text. In the encryption algorithm, following are four different round **transformations: SubBytes, ShiftRow, MixColumn and AddRoundkey [2]. The** first and last rounds differ from other rounds as there is an additional AddRoundKey transformation at the beginning of the first round and no Mix Coulmns transformation is present in the last round.

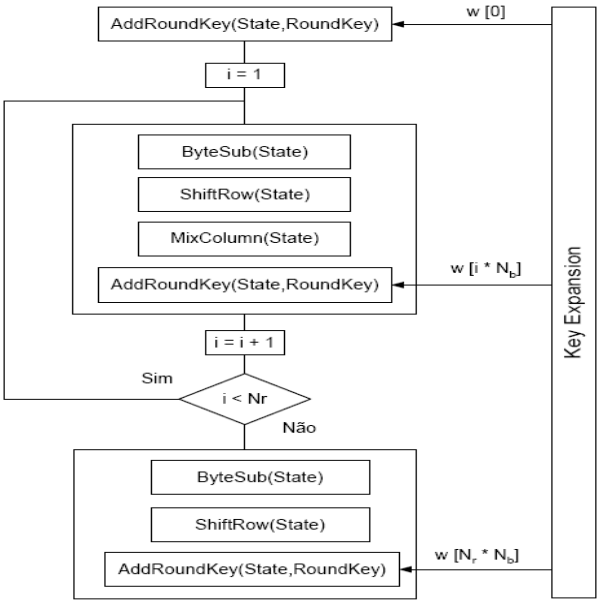

Figure 1: Encryption process for 128 bit key

2.1.1 SubBytes Transformation:

The Sub Bytes transformation is a non-linear byte substitution method and operates on each of the state bytes independently. The Sub Bytes transformation is done using a pre calculated substitution table called as S-box. That S-box table is having 256 numbers (from 0 to 255) and their corresponding resulting values. Using S- box, each byte xy (in hexadecimal) in the matrix is substituted with another byte by looking for the entry in the x-row and the y-column of the table.

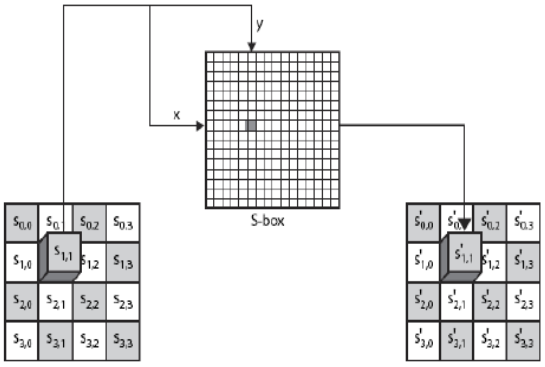

Figure 2: SubBytes transformation process

Table 2: S-Box Table

X \ Y	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	63	7c	77	7b	f2	6b	6f	c5	30	1	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	4	c7	23	c3	18	96	5	9a	7	12	80	e2	eb	27	b2	75
4	9	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5	53	d1	0	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6	d0	ef	aa	fb	43	4d	33	85	45	f9	2	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	6	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	8
c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d	70	3e	b5	66	48	3	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

2.1.2 ShiftRows Transformation:

In Shift Rows transformation method, the rows of the state matrix are cyclically left shifted over different offsets [3]. Row 0 is not shifted by any value; row 1 is shifted by one byte to the left; row 2 is shifted by two bytes to the left and row 3 is shifted by three bytes to the left as shown in following figure.

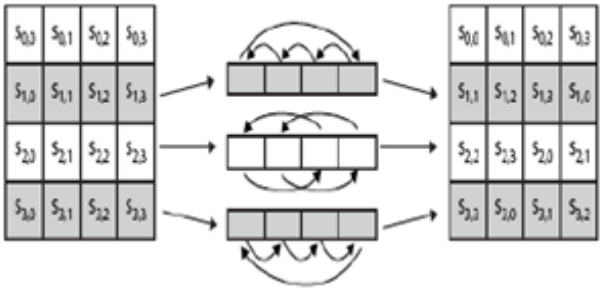

Figure 3: ShiftRow transformation process

2.1.3 MixColumns Transformation:

In MixColumns transformation process, the columns of the state are considered as polynomials over GF (2^8) and multiplied by modulo $x^4 + 1$ with a fixed polynomial given by $c(x)$,

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

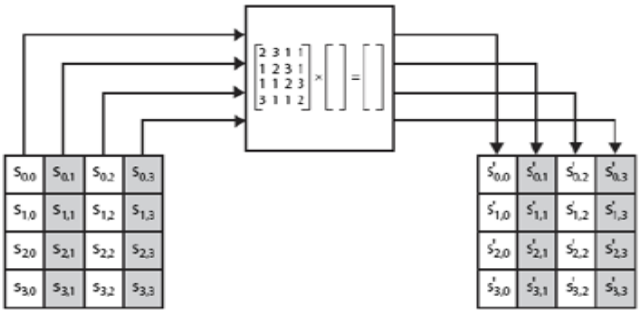

Figure 4: MixColumn Transformation

2.1.4 AddRound Key Operation

The XOR operation is performed between the state and the round key that it is generated from the main key by the Key Generation method. The matrix of keys is represented by w columns. Add Round Key is used both in the encryption and

decryption algorithms [4]. The XOR operation is conducted on byte basis, where the new output byte $S'_{i,j}$ is given by

$s'(i, j) = s(i, j) \oplus w(i, j)$

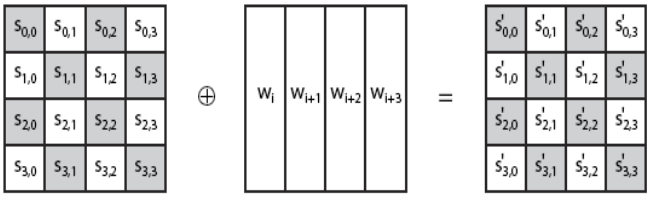

Figure 5: .AddRound Key transformation process

2.2 Decryption Process

Decryption method is a reverse of encryption method that is inverse round transformations for determining the initial plaintext of an encrypted cipher-text in reverse order. The last round of the encryption becomes the first round in decryption process and the expanded key generated in KeyExpansion() is fed back instead of cipher key. The round transformation of decryption uses the subsequent four transformations: AddRound Key, Inv MixColumns, Inv Shift Rows, and Inv SubBytes.

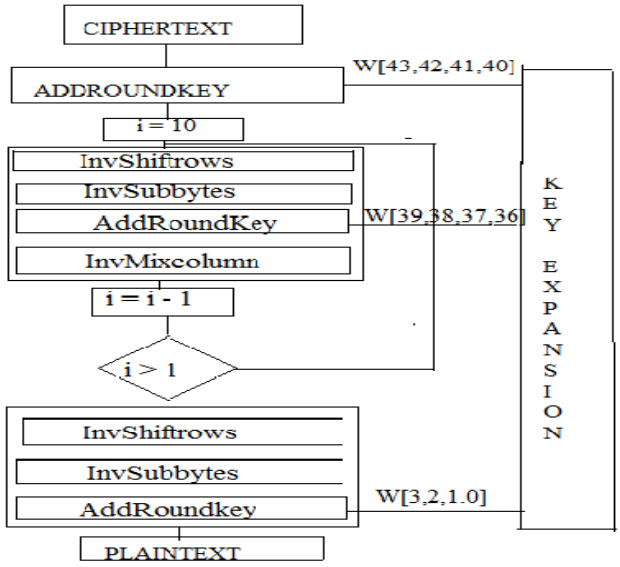

Figure 6: Decryption process for 128 bit key

2.2.1 Inv SubBytes transformation

The Inv SubBytes transformation is done using a once-pre calculated substitution table called as Inv S-box able. Inv S-box table is having 256 numbers (from 0 to 255) and their corresponding values. Inv S-box is presented in following Table III.

Table 3: Inv S-Box Table

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

2.2.2 Inv ShiftRows Transformation

The InvShiftRows() operation is equal to the ShiftRows operation, only the shift is to the right instead of to the left. The first row is not shifted by any value, while the second, third and fourth rows are shifted right by one, two and three bytes respectively.

2.2.3 Inv MixColumn Transformation:

In Inv MixColumn transformation process, the columns of the state are considered as polynomials over GF (2^8) and multiplied by modulo $x^4 + 1$ with a fixed polynomial given by $c(x)^{-1}$,

$c(x)^{-1} = \{0b\}x^3 + \{0d\}x^2 + \{09\}x + \{0e\}$

2.2.4 AddRound Key Operation:

Add Round Key is its own inverse function as the XOR function is having its own inverse value.

3. FPGA And Development Tools

In this project the FPGA kit ALTERA DE2 is used. This board has an EP2C35F672C6 FPGA, from Cyclone II family [8]. The most project package used is that the ALTERA Quartus II 8.1 web Edition. The SoPC (System on Programmable Chip) builder is one among the system-level design supported by the Quartus II software. It is included in the Quartus II software, and allows engineers to rapidly design and evaluates system-on-chip architecture. It is composed of components such as CPU, memory, timer, standard and user-define peripheral, UART, JTAG, Avalon tri-state bridges etc., and also provide an interface to user-defined logics. Additionally, SOPC Builder has in its library the NIOS II Processor, a 32-bit processor to be enclosed within the style. Each component included in the SoPC design has a corresponding PTF file, and this file is used in generating the source code, software component and simulation files for the system.

Once the system is already completed, SOPC Builder generates the optimized bus, and exports the complete system to the Quartus II. Once the hardware project is complete, the program compiles all diagrams and outlines archives and

interprets into connections within the FPGA, then the user will simulate the practicality of his/her project by using an input vector and analysing the output results [3].

4. System Design Using NIOS II

NIOS II is a soft-core processor which is reconfigurable unlike microcontroller. Soft core means the processor core is not fixed in silicon and can be targeted to any ALTERA FPGA family. Reconfigurable means that one can add or remove features to meet performance or price goals. The system design consists of a 32 bit NIOS II processor core, a set of on chip peripherals like JTAG UART to communicate with host PC through the USB cable, 16 KB SRAM that is used by the CPU as a working memory, PIO parallel input output ports, LED and SWITCHES as one of the input output checking device [5] [6].

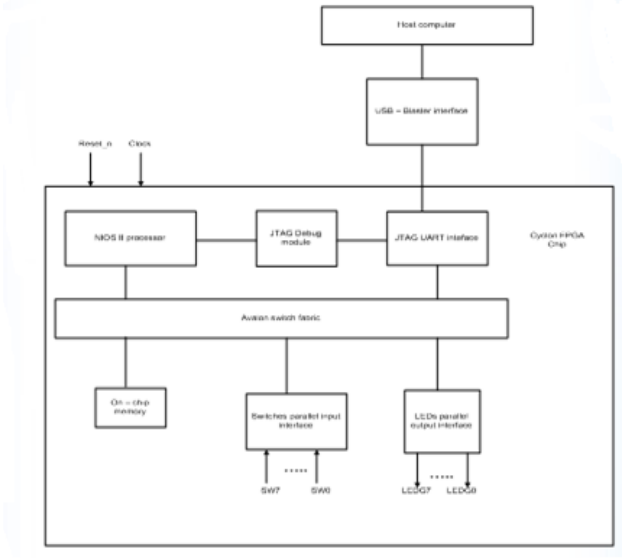

Figure 7: Block diagram of System

The whole system is generated using SOPC Builder tool in Quartus II environment. SOPC Builder automatically generates the interconnect logic to integrate the components in the hardware system [3]. The following figure 8 shows the different component used to generate the system.

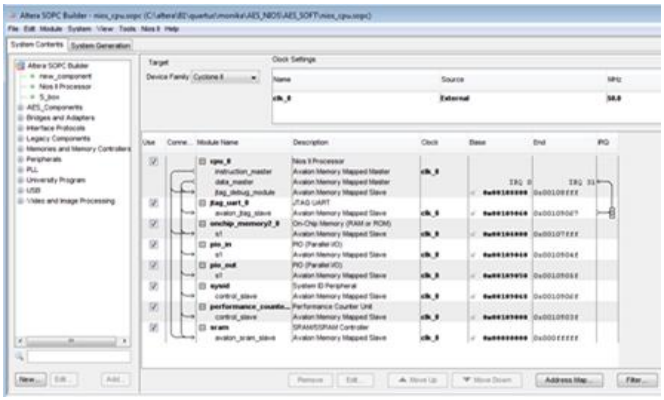

Figure 8: System generated in SOPC Builder

The 'Quartus II' software is used to perform all the required tasks to create the final FPGA hardware design. The compilation of the Quartus II project is done to produce a '.sof' file so as to configure the FPGA. In figure 9, integration of SOPC Builder with Quartus Software is shown. In this BDF the pin assignment is done by importing the pin assignment of Cyclone II (EP2C35F672C6) FPGA [4].

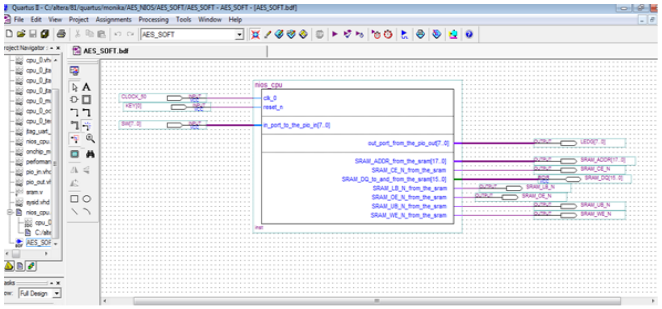

Figure 9: NIOS System without custom instruction Block Diagram File view

5. System Implementation and Results

The encryption and decryption algorithms were successfully run by interconnecting through the GPIO and the results were obtained on the console window in NIOS II IDE. The C code for AES algorithm is written in adsad.c file (Encryption and Decryption separately). Then the project is build using the Build Project command in NIOS II IDE. After the project build the code will be implemented on CYCLONE II (EP2C35F672C6) FPGA using command Run as NIOS II Hardware. After all these steps the output will be seen in NIOS II Console Window as shown in following figure 10. With the performance counter, we can accurately measure execution time taken by multiple sections of code. We need to add only a single instruction at the beginning and end of each section to be measured [7].

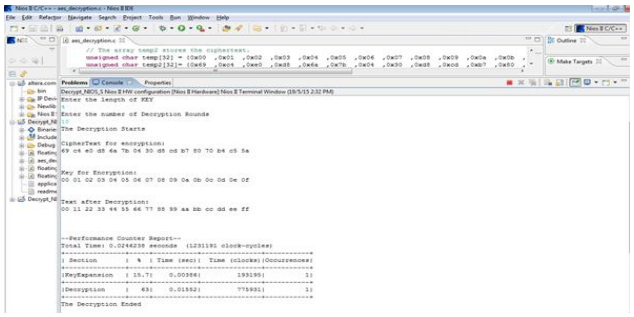

Figure 10: Encryption output for 128 bit key length

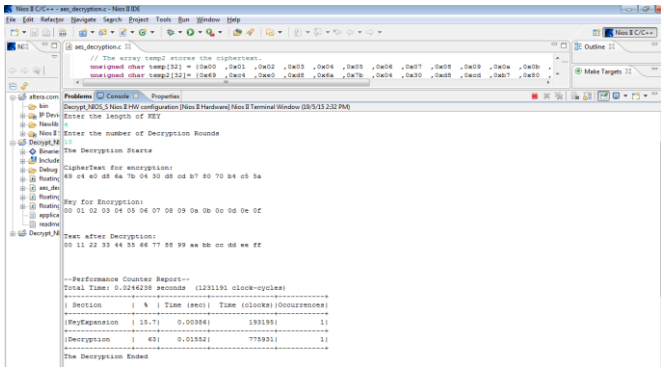

Figure 11: Decryption output for 128 bit key length

As we know in Rijindael's AES data length is always 128 bit but the key length maybe 128, 192 or 256 bit, the different

encryption and decryption output for different key lengths using Nios/e, Nios/s and Nios/f are as follows.

Table 4: Resources used for 128 bit key length encryption

Type of NIOS Core	Logic Cells	Dedicated Logic Registers	Utilization	Time (sec)
NIOS/s	3184	1918	10%	0.0221193
NIOS/e	2372	1327	7%	0.079987
NIOS/f	3898	2334	12%	0.0124165

Table 5: Resources used for 192 bit key length encryption

Type of NIOS Core	Logic Cells	Dedicated Logic Registers	Utilization	Time (sec)
NIOS/s	3184	1918	10%	0.0250258
NIOS/e	2372	1327	7%	0.0904291
NIOS/f	3898	2334	12%	0.0143312

Table 6: Resources used for 256 bit key length encryption

Type of NIOS Core	Logic Cells	Dedicated Logic Registers	Utilization	Time (sec)
NIOS/s	3184	1918	10%	0.0301365
NIOS/e	2372	1327	7%	0.1093
NIOS/f	3898	2334	12%	0.017021

Table 7: Resources used for 128 bit key length decryption

Type of NIOS Core	Logic Cells	Dedicated Logic Registers	Utilization	Time (sec)
NIOS/s	3185	1918	10%	0.0246238
NIOS/e	2377	1327	7%	0.097195
NIOS/f	3890	2334	12%	0.0143355

Table 8: Resources used for 192 bit key length decryption

Type of NIOS Core	Logic Cells	Dedicated Logic Registers	Utilization	Time (sec)
NIOS/s	3185	1918	10%	0.028034
NIOS/e	2377	1327	7%	0.111299
NIOS/f	3890	2334	12%	0.0161768

Table 9: Resources used for 256 bit key length decryption

Type of NIOS Core	Logic Cells	Dedicated Logic Registers	Utilization	Time (sec)
NIOS/s	3185	1918	10%	0.0336718
NIOS/e	2377	1327	7%	0.134075
NIOS/f	3890	2334	12%	0.0194724

6. Conclusion

Using Nios II processor, Rijindael's encryption and decryption algorithm is implemented on ALTERA DE2 educational board, which has EP2C35F672C6 FPGA. The system which is generated using SOPC Builder is compiled in Quartus II software. The hardware required for generation of system is depends upon the logic cells used in CYCLONE II (EP2C35F672C6) FPGA. The performance of the system i.e the time required for the algorithm is calculated by using performance counter for the Nios II/e, Nios II/f and Nios II/f. Different resources used for different Nios II cores are presented with Quartus II 8.1. We can also increase the performance of the system using custom hardware. I can say the performance is increased by comparing the results in the following paper.

Paper	Clock Cycles	Time Required
[4]	7395826	0.147917 sec
This Project	650407	0.01301 sec

References

[1] Dr. Tariq Jamil, "The Rijindael Algorithm" Department of Electrical and Computer Engineering, Sultan Qaboos University (Oman). 0278-6648/04 2004 IEEE

[2] Shunwen Xiao, Yajun Chen, PengLuo, "The Optimized Design of Rijindael Algorithm Based on SOPC", College of Physics and Electronic information China West Normal University, Nanchong, China, 978-0-7695-3922-5/09 2009 IEEE

[3] Meghana A. Hasamnis, Shri Ramdeobaba college of Engg and Management, S. S. Limaye, Jhulelal Institite of Technology,
"Custome Hardware Interface using NIOS II Processor through GPIO", department of Electronics Engg., Nagpur, India, 978-1-4577-2119-9/12/ 2011 IEEE

[4] Madhav M. Deshpande, Meghana A. Hasamnis, "Design of Encryption System using NIOS II Processor", Electronics Department, R.C.O.E.M, Nagpur University, International Journal of Computer Applications, Volume 68- No. 21, April 2013

[5] Altera Corporation, A Refrence Manual on Nios Embedded Processor, 32-Bit Programmer's Online Document: January 2003, Available:http://www.altera.com/literature/manual/mnl_nios_programmers32.pdf.

[6] Altera Corporation, A Tutorial on Nios II Hardware Development, Altera Corporation Website: www.altera.com, June 2006.

[7] Performance counter core, Quartus II Handbook version 9.0 Volume 5: Embedded Peripherals

[8] DE2 Development and Education Board, Altera Corporation