Industrial Engineering Journal
ISSN: 0970-2555
Volume : 53, Issue 11, November : 2024

FPGAIMPLEMENTATION OF A HIGH SPEED AND LOW AREA AES ARCHITECTURE

Gottam Mounika[1] | P Chandhra Sekhar[2]
[1]PG Scholar, Dept. of ECE, Quba College of Engineerig, Nellore.
[2]Assistant Professor, Dept. of ECE, Quba College of Engineering, Nellore.
mounikamow13@gmail.com[1], sekharp21@gmail.com[2]

Abstract

A crucial component of data storage and communication is data security. In a variety of industries, including the military and medical, cryptographic algorithms have been used to offer high-level security against all types of unwanted accesses. Globally, the most secure cryptographic algorithm is the Advanced Encryption Standard (AES), a symmetric cryptographic algorithm. The original architecture was proposed by two Belgian researchers, Joan Daemen and Vincent Rijment, and underwent several modifications. The current changes are meant to improve speed and security. This work suggests a pipelined architecture that is effective in reducing the propagation delay to generate the necessary sub keys during the key expansion process. The key expansion portion of the AES architecture also uses the fork and join architecture in addition to the pipeline structure, which greatly shortens the time needed to generate the necessary subkeys.

I. INTRODUCTION

The society is becoming more and more digitalized therefore. Information security is becoming more important than ever. The need for each individual to identify them self in a digital way has spawn eda wide variety of challenges, such as, how to avoid fraud. Biometric data as finger interior is scan is one way of identification, however in order to use the data that is reliable for identification purposes the data must stay confidential, for that reason information security is important. The biometric data is typically sampled by a physical terminal and the data is transmitted to a centralized server through a unsecured network for verification, in order to protect the data is encryption needed as soon as possible in the data path thus in the terminal. The physical terminal that samples the biometric data must have enough computational power to encrypt the data fast and reliable in a cost-efficient way. Further low power consumption is a requirement for hand held terminals. However biometric data can be rather large, e.g. a passport image with the resolution 3300 x 4400 is 42.5MB uncompressed or 14.2MB with lossless JPEG compression.

The problem of implementing encryption with image application has puzzled research over the last decade. The encryption is often required to be real time yet the processing cannot be do neat a central server. The literature provides several suggestion show to overcome this problem, e.g. partial encryption of the fingerprint. However, this approach might not be feasible for iris scan or voice recognition and complete encryption could be necessary to have sufficient security.

There are clear advantages by using a standard encryption algorithm for the image application. The reliability is well tested and the data can be shared between different platforms encryption standards such as DES (Data Encryption Standard),3DES (Triple Data Encryption Standard) and AES (Advanced Encryption Standard)are standardized by National Institute of Standards and Technology. However, the mentioned encryption algorithms require many

calculations steps and storage of partial results in order to encrypt the data. Therefore is a CPU (Central Processing Unit) not ideal for this type of tasks, since it require many cycles for the CPU toper for them encryption calculations. Previous studies have showed that encryption can be performed much fast using a FPGA (Field Programmable Gate Array) compared to a CPU.

The CPU is a dedicated logical circuit designed to execute instruction and calculation in a sequential order. The FPGA is a programmable logic circuit which means that is function of is not fixed after the silicon fabrication. The device consists of thousands of "building block" called CLB (configurable logic blocks), each of CLB can be individually be configured to a specific logic function and each of the CLB can(with some limitation) be connected to any other CLB through the routing network. A typical FPGA architecture, the CLB slices are located in a matrix pattern and is surrounded by several different types of dedicated blocks; multipliers, Block random access memory (BRAM) and digital controlled clocking managers (DCM). The configuration of the components and their interconnectivity implements the actual functionality of the FPGA.

Implementation of AES128 bit algorithm on FPGA boards for Alter a devices uses VHDL code. At first it was recommended by Rijndael on Oct-2000. Cryptography is the art and science of protecting information from undesirable individuals by converting it into a form non- recognizable by its attackers while stored and transmitted. The aim of this project is to reduce the time, area and power consumption. In order to implement the pipelined architecture is used to speed up the processing and run at very high speed. It is possible by introducing lookup tables instead of multipliers. And it is possible to construct 196 and 256 bit AES Algorithm. But it leads very complex circuitry and low end is very applicable in the present market. The AES algorithm is implemented in FPGA (Field Programmable Gate Array) for ease of operation in increased frequency. This project utilizes the Quartus II software simulation tool and synthesizer.

Quartus II Simulator too lisenas lo guest Electronic Design Automation(EDA) and is similar to ECAD Software. Category of Standard is Information Security Standard, Cryptography. Explanation is about The Advanced Encryption Standard(AES) specifics FIPS-approved cryptographic algorithm that can be used to protect electronic data. The AES algorithm is a symmetric block cipher that can encrypt (encipher) and decrypt (decipher) information. Encryption converts data to an unintelligible form called ciphertext; decrypting the cipher text converts the data back into its original form, called plaintext and is implemented for high-speed devices of altera in order to obtain high debugging options. It is very compact to find whenever the problem occurs. The AES algorithm is capable of using cryptographic keys of 128, 192, and 256 bits to encrypt and decrypt data in blocks of 128 bits. In this project 128-bit input and cipher key size is used. Applicability is all online services are using this algorithm for Example online State Bank of India. The strength of an FPGA compared to a CPU is that many smaller circuits can be implemented to run in parallel, while the CPU is a native sequential circuit. The partial results for the encryption can be calculated in parallel and combined later stage, reducing the number of cycles required significantly. The FPGA cannot run at the same clock frequencies as a modern CPU (which is in the GHz range), the FPGA runs typically with an internal processing clock of 250-2500 MHz

Despite the lower clock frequency an FPGA process data much faster than a CPU, due to parallel processing capabilities, thus encrypts data at a high rate. The ideal

technology for implementing AES in hardware is ASIC (Application specific integrated circuit). Here is it possible to custom design your chip to the application and achieving speeds that are superior to both CPU and FPGA. However, the cost of design an ASIC far exceeds the scope of this project both in development time and production cost. Therefor is the second-best choice the FPGA. This AES algorithm was implemented in so many languages like C, C++, JAVA and VHDL. In this project it is implemented in VHDL code.

1.1 BLOCKDIAGRAM FOR AES STREAM CIPHER

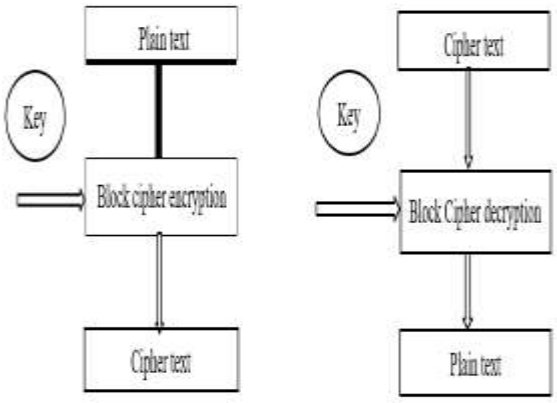


Figure1.1 AES block diagram of encryption and decryption

1.2 BLOCK

AES is a block cipher. This means that the number of bytes that it encrypts is

fixed.AEScancurrentlyencryptblocksof16bytes atatime;nootherblocksizesarepresentlya part of the AES standard. If the bytes being encrypted are larger than the specified block then AES is executed concurrently. This also means that AES has to encrypt a minimum of 16 bytes. If the plain text is smaller than 16 bytes then it must be padded. Simply said the block is a reference to the bytes that are processed by the algorithm.

1.3 STATE

Defines the current condition (state) of the block that is the block of bytes that are currently being worked on. The state starts off being equal to the block, however it changes as each round of the algorithms executes. Plainly said this is the block in progress. **XOR** Refers to the bitwise operator **Exclusive OR, XOR** operates on the individual bits in a byte in the following way:

0XOR 0 =0,

1XOR 0=1,

1XOR 1 =0,

0XOR 1 =1

For example, the Hex digits D4XORFF1101010XOR11111111=00101011 (Hex2B) Another interesting property of the XOR operator is that it is reversible. So Hex 2BXORFF = D4. Most programming languages have the XOR operator built in.

HEX

It defines a notation of numbers in base 16. This simply means that; the highest number that can be represented in a single digit is 15, rather than the usual 9 in the decimal (base10) system.

Table1.1 Hex to Decimal

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
2	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
3	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
4	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
5	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
6	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111
7	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
8	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143
9	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159
A	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175
B	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191
C	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207
D	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223
E	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239
F	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255

Industrial Engineering Journal
ISSN: 0970-2555
Volume : 53, Issue 11, November : 2024

For example, using the above table 1.1 HEX D4 = DEC 212. All of the tables and examples are written in HEX. There as on for this is that a single digit of Hex represents exactly 4 bits. This means that a single byte can always be represented by HEX digits. This also makes it very useful in creating lookup tables where each HEX digit can represent a table index.

II. LITERATURESURVEY

This project will focus on implementation AES in a FPGA which is a standardized algorithm that is recognized by the literature. The image data for encryption is biometric samples e.g. finger print images. The chosen architectures Xilinx Kintex 27,since this FPGA family is the market leader on performance versus power versus price. The Kintex27 utilizes 28 nm die technology, which is minimize the dynamic power consumptions compared to previous 40 nm die technology. The implementation should be portable to similar architectures, thus be compatible with Artix27 and Virtex27. The project includes performance measurements of the implementation, in respect to encryption speed, power consumption and data integrity.

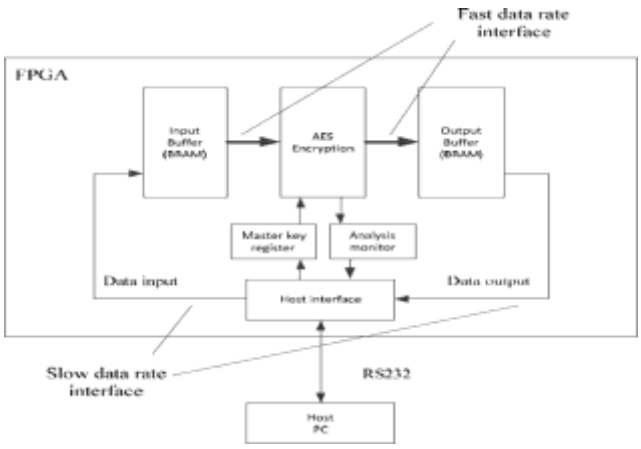


Figure2.1 Block Diagram

The host starts by loading the master key into the master key register. The hosts ends the test data to the host interface through RS232 (Serial single2ended data and control interface).The host interface loads the data block into the input buffer.

The AES starts the encryption when a block of data is ready in the buffer. The data is loaded through a wide fast interface to avoid band width problems. The AES encrypts the data and sends the result to the output buffer, the host interface transfers the finished block back to the PC for integrity analysis. The analysis monitor sends the performance statistics back to the PC for further analysis.

HIGH PERFORMANCE AES FPGA IMPLANTATIONS

Rahimun is a et al describing the Parallel sub pipelined (PSP) architecture. The PSP architecture uses 128 bit data blocks which are divided into four blocks of 32 bit, each of these 32 bit blocks are process Edina parallel, in order to achieve high through put. The architecture is a mix of parallel and sequential processing, which has achieved a high efficiency. The design has been both implement Edina Virtex 26LX75T FPGA and prototype dasan ASIC design. The through put achieved on Virtex 26LX75T was 59.59Gb/s, the area used was 2597 slices, giving an efficiency of 22.94 Mb/slice. The results was retrieved be simulating the design using Model Sim (VHDL/Verilog simulator). The work included power simulations for130 nm and 180 nm ASIC die technology.

[27]Liu, Xu and Yuan published real time AES encryption was in focus. The paper describes a 66.1Gb/s fully pipelined AES 128bit FPGA implementation. The FPGA was implemented on the new Xilinx Virtex27 VX690T device, they achieved 66.1 Gb/s using 3436 slices thus achieving an efficiency of 19.20 Mb/Slice. The latency of the design is 22 clock cycles at a clock running at 516

Industrial Engineering Journal
ISSN: 0970-2555
Volume : 53, Issue 11, November : 2024

MHZ, which is equal to 426 us. The paper further suggest to run two AES kernels in order to break the 100Gb/s barrier, this should be possible with the chosen target, since only a fraction of the slices are used. The design was only simulated and no power estimations were performed.

[9]This FPGA implementation of AES encryption as counter mode for 256 bits data width was done by Balwinder Singh, Harpreet Kaur and Himanshu Monga in 2010.They achieved to encrypt at 52.6124 Gbit/s with a master key length of 256 bits. The design was implemented in Xilinx Spartan 3, Xilinx Virtex II and Xilinx Virtex E devices.

Manoj and Manjula implemented AES 128 bit as an image application in a Xilinx Spartan 3 device in 2012. The design was similar to what others have done, expect that the design could take 8 bit input (data pixels) and unroll them to 128 bit, which is a trivial task. The encryption throughput was 882.46 Mb/s, the efficiency was 0.53Mb/slice and the latency was 24 clocks. The article includes plots with the relationbetweencorevoltageandpowerconsumptionsforthedevice.However,thesedrawingswere not commented and no power estimate of the design was made.

The literature review has revealed a knowledge gap so far there has been little focus on power consumption. The reason could be that the published designs are only conceptual and the problem of reducing power consumption is left out for further research.

III. PROPOSEDMETHOD

AES ALGORITHM

AES as well as most encryption algorithms is reversible. This means that almost the same steps are performed to complete both encryption and decryption in reverse order. The AES algorithm operates on bytes, which makes its impler to implement and explain. This key is expanded into individual sub keys, a subkeys for each operation round. This process is called Encryption.

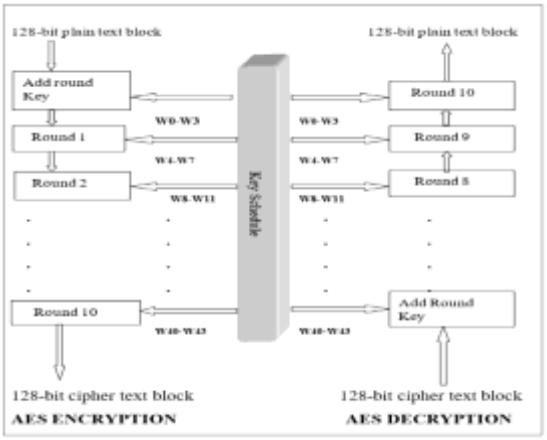


Figure3.1 Internal Structure of AES

AES ENCRYPTION

Each round of processing includes one single-byte based substitution step, a row-wise permutation step, a column-wise mixing step, and the addition of the round key. The order in which these four steps are execute dis different for encryption and decryption.

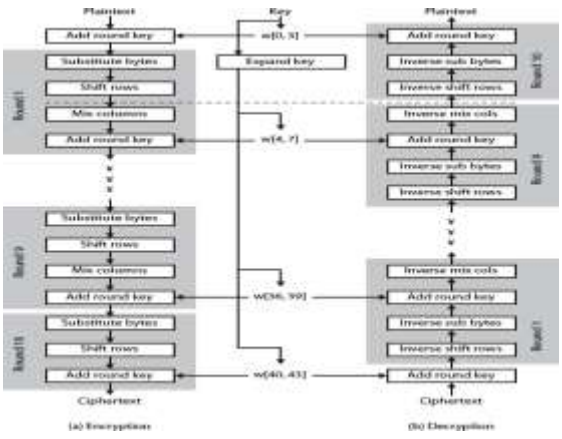
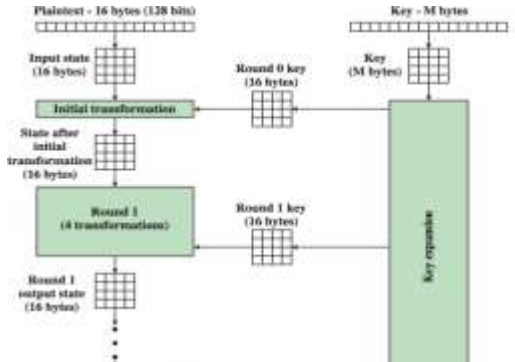


Figure3.2 Structure of AES



Industrial Engineering Journal
ISSN: 0970-2555
Volume : 53, Issue 11, November : 2024

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
XOR	XOR	XOR	XOR	XOR	XOR	XOR	XOR	XOR	XOR	XOR	XOR	XOR	XOR	XOR	XOR
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

Figure3.3 AES Encryption

ADD ROUND KEY

- Takes 128-bit (16-byte) key and expands into array of 44/52/6032-bitword
- Start by copying key into first 4 words Then loop creating words that depend on values in previous & 4 places back in 3 of 4 cases just XOR the set together.
- 1st word in 4 has rotate + S-box + XOR round constant on previous, before XOR 4thback. The goal of the substitution step is to reduce the correlation between the input bits and the output bits at the bit level.

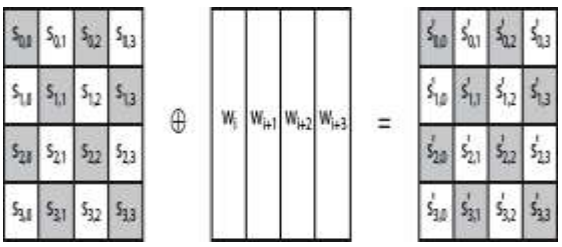


Figure3.4 Add Round Key

Each of the 16 bytes of the state is XORed against each of the 16 bytes of a portion of the expanded key for the current round. The Expanded Key bytes are never reused. So once the first 16 bytes are XOR ed against the first 16 bytes of the expanded key the n the expanded key bytes 1-16 are never used again. The next time the Add Round Key function is called bytes 17-32 are XOR ed against the state.

The first time Add Round Key gets executed State

Table3.1AddRoundKey

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
XOR	XOR	XOR	XOR	XOR	XOR	XOR	XOR	XOR	XOR	XOR	XOR	XOR	XOR	XOR	XOR
17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32

The second time Add Round Key is executed

Table3.2Add Round Key

And so on for each round of execution. During decryption this procedure is reversed. Therefore the state is first XOR ed against the last16 bytes of the expanded key, then the second last 16 bytes and so on. Theme trod for deriving the expanded key is described.

SUB BYTES

The **Sub Bytes()** transformation is a non-linear byte substitution that operates independently on each byte of the State using a substitution table (S-box). This Show which is invertible, is constructed by composing two transformations S-BOX During encryption each value of the state

Industrial Engineering Journal
ISSN: 0970-2555
Volume : 53, Issue 11, November : 2024

is replaced with the corresponding S-BOX value.

AESS-BoxLookupTable.

Table3.3 Substitution Box

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
)	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
!	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
)	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

For example: -HEX 19 would get replaced with HEXD4.

During decryption each value in the state is replaced with the corresponding inverse of the S-BOX.

Table3.4InverseS-Box

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	52	09	6A	D5	30	36	A5	38	VF	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	08	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	00	8C	BC	D3	0A	F7	E4	58	05	B8	B3	45	06
7	D0	2C	1E	8F	CA	3F	0F	02	C1	AF	VD	03	01	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	1B
B	FC	56	3E	4B	C6	D2	79	20	9A	DB	C0	FE	78	CD	5A	F4
C	1F	DD	A8	33	88	07	C7	31	B1	12	10	59	29	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	DB	BB	3C	83	53	99	61
F	17	2B	04	7E	BA	77	D6	26	E1	69	14	63	55	21	0C	7D

For example HEX D4would get replaced with HEX19

SHIFT ROW

A circular byte shift in each

1st row is un changed

2nd row does 1 byte circular shift to left

3rd row does 2 byte circular shift to left

4th row does 3 byte circular shift to left

Decrypt inverts using shifts to right

Since state is processed by columns, this stepper mutes bytes between the columns

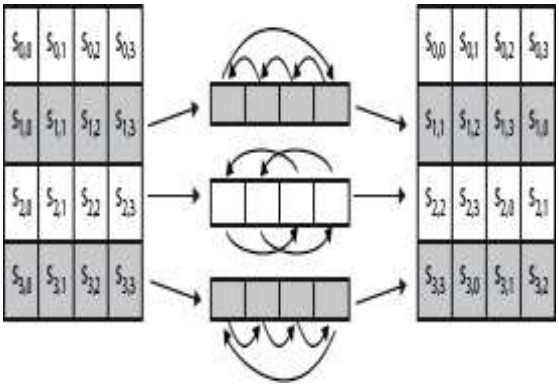


Figure3.5 Shift Row

Arranges the state in a matrix and then performs a circular shift for each row. This is not a bitwise shift. The circular shift just moves each byte one space over. A byte that was in the second position may end up in the third position after the shift. The circular part of it specifies that the byte in the last position shifted one space will end up in the first position in the same row. The state is arranged in a 4x4 matrix (square).

byte0	**byte4**	**byte8**	**byte12**
byte1	**byte5**	**byte9**	**byte13**
byte2	**byte6**	**byte10**	**byte14**
byte3	**byte7**	**byte11**	**byte15**

The confusing part is that the matrix is formed vertically but shifted horizontally. So the first 4 bytes of the state will form the first bytes in each row.

Industrial Engineering Journal
ISSN: 0970-2555
Volume : 53, Issue 11, November : 2024

So bytes 1 2 3 45 6 7 8 9 10 11 12 13 14 15 16

Will form a matrix:

1	5	9	13
2	6	10	14
3	7	11	15
4	8	12	16

Each row is then moved over (shifted) 1,2or3 spaces over to the right, depending on the row of the state. First row is never shifted

Row10

Row 21

Row 32

Row 43

The following table show show the individual bytes are first arranged in the table and then move dover (shifted).

Blocks 16 bytes long:

From				To			
1	5	9	13	1	5	9	13
2	6	10	14	6	10	14	2
3	7	11	15	11	15	3	7
4	8	12	16	16	4	8	12

During decryption the same process is reversed and all rows are shifted to the left

From				To			
1	5	9	13	1	5	9	13
2	6	10	14	14	2	6	10
3	7	11	15	11	15	3	7
4	8	12	16	8	12	16	4

MIXCOLUMN

The **Mix Columns()** transformation operates on the State column-by-column, treating each column as a four-term polynomial.

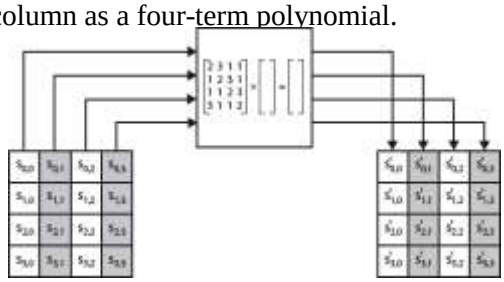


Figure3.6 Mix Column

This is perhaps the hardest step to both understand and explain. There are two parts to this step. The first will explain which parts of the state are multiplied against which parts of the matrix. The state is arranged into a 4 row table (as described in the Shift Row function).The multiplication is performed one column at a time (4bytes). Each value in the column is eventually multiplied against every value of the matrix (16total multiplications). The results of these multiplications are XORed together to produce only 4 result bytes for the next state. Therefore 4 bytes input, 16 multiplications 12XORs and 4 bytes output. The multiplication is performed one matrix row at a time against each value of a state column.

AESDECRYPTION

For decryption, each round consists of the following four steps

INVERSESHIFTROWS

DuringdecryptionisdenotedInvShiftRowsforInverseShiftRowsTransformation.The goal of this transformation is to scramble the byte order inside each 128-bit block. For decryption, the corresponding step shifts the rows in exactly the opposite fashion. The first row is left unchanged, the second row is shifted to the right by one byte, the third row to the right by two bytes, and the last row to the right by three bytes, all shifts being circular.

INVERSESUBSTITUTEBYTES

The corresponding substitutions tepused during decryption is called Inv Sub Bytes.

ADDROUNDKEY

The corresponding step during decryption is denoted In v Add Round Key for inverse add round key transformation

Industrial Engineering Journal
ISSN: 0970-2555
Volume : 53, Issue 11, November : 2024

INVERSEMIXCOLUMNS

The corresponding transformation during decryption is denoted In v Mix Columns and stands for inverse mix column transformation. The goal is here is to further scramble up the 128-bit input block. The third step consists of XORing the output of the previous two steps with four words from the key schedule. Note the differences between the order in which substitution and shifting operations are carried out in a decryption round vis-a-vis the order in which similar operations are carried out in an encryption round. The last round for encryption does not involve the "Mix columns" step. The last round for decryption does not involve the "Inverse mix columns" step.

During decryption the Mix Column the multiplication matrix is changed to:

Other then the change to the matrix table the function performs the same steps as during encryption.

IV. AES KEY EXPANSION ALGORITHM

Assuming a 128-bit key, the key is also arranged in the form of a matrix of 4 × 4bytes. As with the input block, the first word from the key fills the first column of the matrix, and so on. Prior to encryption or decryption the key must be expanded. The expanded key is used in the **Add Round** Key function defined above. Each time the Add Round Key function is called a different part of the expanded key is XOR ed against the state. In order for this to work the Expanded Key must be large enough so that it can provide key material for every time the Add Round Key function is executed. The Add Round Key function gets called for each round as well as one extra time at the beginning of the algorithm.

k0	k4	k8	k12
k1	k5	k9	k13
k2	k6	k10	k14
k3	k7	k11	k15

Therefore the size of the expanded key will always be equal to:16 * (number of rounds + 1). Each round has its own round key that is derived from the original 128-bitencryption key in the manner described in this section. One of the four steps of each round, for both encryption and decryption, involves XORing of the round key with the state array. The AES Key Expansion algorithm is used to derive the 128- bit round key for each round from the original 128-bit encryption key. As you'll see, the logic of the key expansion algorithm is designed to ensure that if you change one bit of the encryption key, it should affect the round keys for several rounds. In the same manner as the 128-bit input block is arranged in the form of a state array, the algorithm first arranges the 16 bytes of the encryption key in the form of a4×4 array of bytes,

The 16 in the above function is actually the size of the block in bytes. This provides key material for every byte in the block during every round+1.

U

$0E$	$0B$	$0D$	09
09	$0E$	$0B$	$0D$
$0D$	09	$0E$	$0B$
$0B$	$0D$	09	$0E$

w0 w1

Industrial Engineering Journal
ISSN: 0970-2555
Volume : 53, Issue 11, November : 2024

w2 w3

The first four bytes of the encryption key constitute the word w0, the next four bytes the word w1, and so on.
The algorithm subsequently expands the words **[w0,w1,w2,w3]** into a 44-word key schedule that can be labelled **w0, w1, w2, w3... w43.**

Of these, the words [w0, w1, w2, w3] are bitwise XOR ed with the input block before the round-based processing begins. The remaining 40 words of the key schedule are used four words at a time in each of the 10 rounds. The above two statements are also true for decryption, except for the fact that we now reverse the order of the words in the key schedule, The last four words of the key schedule are bitwise XOR ed with the128-bit cipher text block before any round-based processing begins. Subsequently, each of the four words in the remaining 40 words of the key schedule are used in each of the ten rounds of processing.

Now comes the difficult part: How does the Key Expansion Algorithm expand four words w0,w1, w2,w3 into the44 words.
w0,w1, w2, w3, w4, w5, ,w43

As shown in the figure, the key expansion takes place on a four-word to four-word basis, in the sense that each grouping of four words decides what the next grouping off our words will be

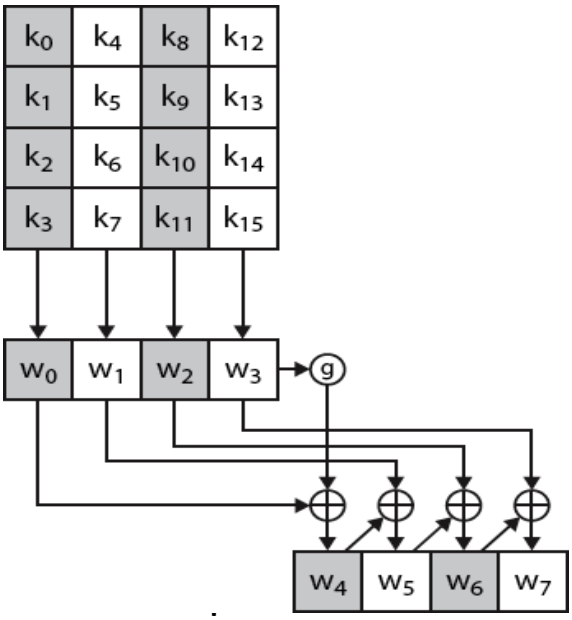


Figure4.1AES Key Expansion

The key expansion takes place on a four-word to four-word basis as shown here

The Algorithmic Steps in Going from a 4-Word Round Key to the Next 4-Word Round Key. We now come to the heart of the key expansion algorithm we talked about in the previous section generating the four words of the round key for a given round from the corresponding four words of the round key for the previous round. Let's say that we have the four words of the round key for the ith round:

FUNCTION

The function(s) that will return the 4 bytes written to the effected expanded key bytes. Notice that most numbers that change in following tables match the current round number. This makes implementation in code much easier as these numbers can easily be replaced with loop variables.

16BYTEKEYEXPANSION

Each round (except rounds 0, 1, 2 and 3) will take the result of the previous round and produce a 4 byte result for the current round. Notice the first 4 rounds simply copy the total of 16 bytes of the key. The functions of Key Expansion are described

REALISATION OF AES ARCHITECTURE

A new architecture for a high speed AES encryption, decryption and combined encryption and decryption using 128-bits key size is presented. This architecture is implemented using fully pipelining method. This new architecture has shown greater performance in terms of throughput and Area comparing to previous pipelined AES Cryptography. The proposed top module of

Industrial Engineering Journal
ISSN: 0970-2555
Volume : 53, Issue 11, November : 2024

AES Encryption is Similar to the existing pipelined designs, the proposed structure first uses loop unrolling to expand the 10-round operations and adds registers between rounds, forming a one pass data path for plain texts. For Decryption the rounds will be 10 to1.

IMPLEMENTATION ASPECTS

It concerns FPGAs(Field Programmable Gate Arrays). The basic FPGA blocks, I/O, CLBs(Combinational Logic Blocks), and routing architecture, are

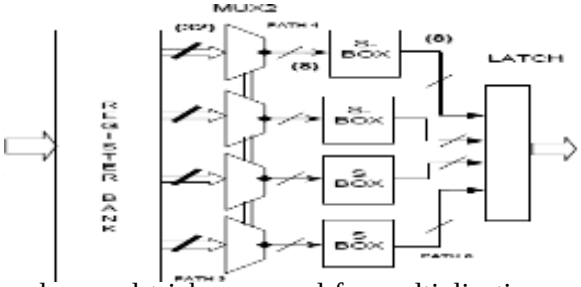
DESIGNFLOW

ENTITY

FITTINGANDASSEMBLING

discussed to impart a basic understanding of FPGA operation. The static RAM implementation method for the programming elements of FPGAs. An in depth investigation of one common static RAM chip ,I/O, CLB, and routing configurations used in industry.

IMPLEMENTATION OF MIX COLUMN IN MIX COLUMN



and several tricks are used for multiplications. The architectural view of mix column. Circuit Architecture of Mix Column. Byte-wise multiplications include multiplying the data by 1, 2, and 3. Multiplying by 1 the data will remain the same. For multiplication by 2, the 8 bit data is left shifted by 1 bit, and the LSB is replaced by 0. Then the MSB of the original data is used for comparison. If it is 0, then the left shifted data is the result; if it is 1, then the left shifted value is XORed with the reduction polynomial, in this case 00011011, to generate the result. For multiplication by 3 the original byte is simply XORed with the result of multiplication by 2. Using the above method, the multiplication by 1, 2, and 3 of each of the bytes in the data are determined. Then the correct combinations of values are XORed with each other to produce a new byte. The same process goes on until allthe16 bytes in the data are replaced.

V. ALTERA QUARTUS II

The Quartus II development software provides a complete design environment for FPGA designs. Design entry using schematics, block diagrams of VHDL, and Verilog HDL. Design analysis and synthesis, fitting, assembling, and timing analysis, simulation

TIMINGANALYSIS

SIMULATION

Figure5.1 Design Analysis

PIPELINEDARCHITECTUREFOREN CRYPTER

The overall pipelined architecture for AES Encryptor looks as shown below. It includes

Figure5.2 Pipelined Architecture

MULTIPLEXER(MUX)

In electronics, a multiplexer (or mux) is a device that selects one of several analog or digital input signals and forwards the selected input into a single line. A multiplexer of 2^n inputs has n select lines, which are used to select which input line to send to the output.

Industrial Engineering Journal
ISSN: 0970-2555
Volume : 53, Issue 11, November : 2024

Multiplexers are mainly used to increase the amount of data that can be sent over the network with in a certain amount of time and band width. A multiplexer is also called a data selector.

Figure5.3 Variable Multiplexer

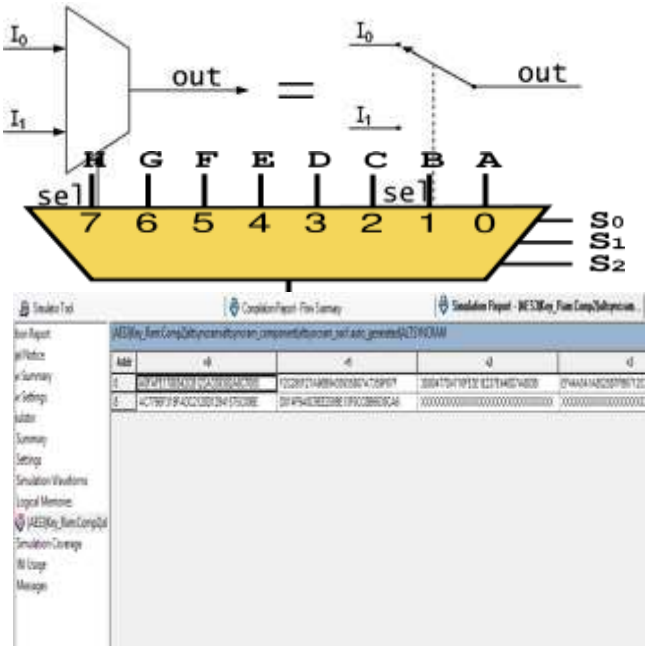


Figure5.4 Variable Mux

MULTIPLIERS

Multipliers play an important role in today's digital signal processing and various other applications. With advances in technology, many researchers have tried and are trying to design multipliers which offer either of the following design targets , high speed, low power consumption, regularity of layout and hence less area or even combination of the min one multiplier thus making them suitable for various high speed, low power and compact VLSI implementation. The common multiplication method is "add and shift" algorithm. In parallel multipliers number of partial products to be added is the main parameter that determines the performance of the multiplier. To reduce the number of partial products to be added, Modified Booth algorithm is one of the most popular

algorithms.

The salient features of the

AES ENCRYPTION/DECRYPTION are summarized in the following manner:

HIGH THROUGH PUT

For fully Pipelined implementation, are a requirements increase with larger Key Size but through put (No. of blocks processed per second) is unaffected.

PARAMETER FLEXIBILTY

Any combination of Key sizes and Block sizes those are multiples of 32 bits can be accommodated. As a result, number of rounds can be modified.

IMPLENTATION FLEXIBILTY

Decryption can be implemented in same structure as Encryption. (Though with different components).

NOKNOWNSECURITYATTACK

Although it has received criticism due to its simple mathematical structure.

VI. RESULTS & DISCUSSIONS

Industrial Engineering Journal
ISSN: 0970-2555
Volume : 53, Issue 11, November : 2024

Figure6.1 Flow Summary

The AES3 program in Stratix 3 FPGA initializes the hardware components for AES encryption/decryption, loads the key, and feeds input data for processing. The FPGA executes the AES algorithm, including key expansion, substitution, permutation, and XOR operations. After processing, the program retrieves the resulting ciphertext or plain text from the FPGA. Finally, it cleans up resources and prepares for subsequent operations.

Figure6.2 Contents of the RAM which stores the AES encrypted version

The RAM storing an AES encrypted version contains the cipher text, which is the encrypted form of the original data. A long side the cipher text, it may store the initialization vector (IV) used during encryption. Additionally, there might be temporary variables, buffers, or keys involved in the encryption process. These contents are temporary and volatile, meaning they are erased when power is removed from the system or when the data is no longer needed.

Figure6.3 Coverage Summary of AES

The coverage summary of AE Sin Quartus II typically includes details on the synthesis, implementation, and timing analysis stages. It assesses how well the design meets specified criteria such as area utilization, performance, and timing constraints. This summary helps ensure the design is optimized for efficient FPGA implementation while meeting the required cryptographic standards and performance targets.

Figure 6.4 Simulation Wave forms which shows the AES Key Generation

Simulation waveforms depicting AES key generation show case the progressive expansion of the initial key into a series of round keys through operations such as key scheduling, round constant addition, and various transformations, providing a visual representation of the key expansion process overtime

Figure6.5 Simulation Waveform which shows the clock and reset

To simulate waveforms for the clock and reset signals of an AES Architecture, in this typically use a hardware description language (HDL) such as Verilog or VHDL along with a simulation tool .

Figure6.6 Compilation Report of AES algorithm

A compilation report on the AES algorithm provides a detailed overview of its design,

implementation, security properties, and real-world applications ,serving comprehensive source for understanding this widely-used encryption standard. It encompasses AES's history, cryptographic operations, security analysis, performance evaluation, and comparisons with other encryption algorithms

VII. CONCLUSION

The present work provides with the basic information needed to implement the AES Encryption/Decryption algorithm for high speed Altera Devices. The mathematics and design reasons behind AES were purposely left out. A FPGA implementation of area-optimized AES algorithm which meets the actual application is proposed in this project. After being coded with VHDL Hardware Description Language, the wave form simulation of the new algorithm was taken in the Quartus platform uses EDA Simulation tool equal to ECAD Tool. Ultimately, a synthesis simulation of the new algorithm has been done. The AES algorithm was realized in the Stratix–II device which aims for high speed and it was achieved.

VIII. FUTURESCOPE

One could work on selection of a larger key size which would make the algorithm is more secure, and a larger input block to increase the through put. The extra increase in area can however be tole rated. So such an algorithm with high level of security and high through put can have ideal applications such as in multimedia communications. Furthermore study of optimization approaches for the implementations supporting multiple key lengths and modes of operation have tremendous scope for future work.

One could require to port on hardware and to be obtained at hardware outputs.

In this project I had the implementation of AES Algorithm. We can extend this implementation to image encryption and decryption using AES algorithm.

Video encryption and decryption has to be done.

Encryption and decryption along with compression in which makes networks speeds Better.

IX. REFERENCES

1. J. Daemen and V. Rijmen, AES Proposal: Rijndael, AES Algorithm Submission, September 3, 1999.
2. J. Daemen and V. Rijmen, The block ciphers Rijndael, Smart Card research and Applications, LNCS1820, Springer-Verlag, pp.288-296.
3. A. Lee, NIST Special Publication 800-21, Guide line for Implementing Cryptography in the Federal Government, National Institute of Standards andTechnology,November1999.
4. A. Menezes, P. vanOor schot, and S. Vanstone, Hand book of Applied Cryptography, CRC Press, New York, 1997, p. 81-83.
5. J. Nechvatal, et. al., Report on the Development of the Advanced Encryption Standard (AES), National Institute of Standards and Technology, October 2,2000.
6. “Advanced Encryption Standard (AES)” Federal Information Processing Standards Publication 197, Nov. 2001
7. http://www.xilinx.com/appnotes/FPGA_NSREC98.pdf.
8. W. Kühnet.al., FPGA based Compute Nodes for High Level Triggering in PANDA, Journal of Physics: Conference Series 119 (2008) 022027.
9. James H. Wiebe, AES-128 implementation on a Virtex-4 FPGA Proc. 2007 IEEE International Symposium on Signal Processing and Information Technology.
10. B. Singh, H. Kaur, and H. Monga FPGA Implementation of AES Coprocessor in Counter Mode proc pp. 491–496, 2010. © Springer-Verlag Berlin Heidelberg2010

11. Song J. Park, "Analysis of AES Hardware Implementations" Department of Electrical& Computer Engineering Oregon State University, Corvallis,
12. PRAVINB. GHEWARI, MRS.JAYMALA, K. PATILAMITB. CHOUGULE," Efficient Hardware Design and Implementation of AES Cryptosystem" International Journal of Engineering Science and Technology Vol. 2(3), 2010,213-219.
13. Panu Hamalainen, Marko Han Niskanen, Timo Hamalainen, and Jukka Saarinen, "HARDWARE IMPLEMENTATION OF THE IMPROVED WEP AND RC4ENCRYPTION ALGORITHMSFORWIRELESSTERMINALS"
14. Rijndael, N.Sklavosando. koufopavlou," architecture and VLSI implementations of the AES-Proposal", .IEEE TRANSACTIONS ON COMPUTERS, VOL. 51,NO. 12, DECEMBER 2002."Utilizing hard cores of modern FPGA devices for high-performance cryptography" Tim Guneysu. J Crypto grEng(2011)1:37–55DOI10.1007/s13389-011-0002-2.
15. A. Mazzeo, L. Romano, G.P. suggese and N.Mazzocca.2003. FPGA Implementation of a Serial RSA Processor. Design. Proceedings of the conference on Design, Automation and Test in Europe-Volume 1.ISBN:0-7695-1870-2.
16. A. J. Elbirt, W. Yip, B. Chetwynd, and C. Paar. An FPGA implementation and performance evaluation of the AES block cipher candidate algorithm finalist. p[Online]. Available: http://csrc.nist.gov/encryption/aes/round2/conf3/aes3papers.html.
17. M. McLoone and J. V. McCanny, "Rijndael FPGA implementation utilizing look-up tables," in IEEE Workshop on Signal Processing Systems, Sept. 2001,pp. 349–360.
18. K.U. Jarvinen, M.T.Tommiska, and J.O.Skytta, "A fully pipelined memoryless 17.8 Gbps AES-128 encryptor," in Proc. Int. Symp. Field Programmable Gate Arrays (FPGA 2003), Monterey, CA, Feb. 2003, pp.207–215.
19. G.P.Saggese, A.Mazzeo, N.Mazocca, and A.G.M.Strollo,"An FPGA based performance analysis of the unrolling, tiling and pipe lining of the AES algorithm," in Proc. FPL2003, Portugal, Sept. 2003.