Design of High Performance 32-bit Cryptographic processor

Manisha.Guduri*, V. Jaya Rajesh**
Department of Electronics & Communications Engineering,M.Tech (VLSI System Design),Assistant Professor in Alamuri Ratnamala Institute of Engineering & Technology.
Department of Electrical Engineering ,M.Tech (Power System),Assistant Professor in Alamuri Ratnamala Institute of Engineering & Technology

Abstract- With rapid increases in communication and network applications, Cryptography has become a crucial issue to ensure the security. In this proposed paper a microcode-based architecture with a novel reconfigurable data path which can perform either prime field operations or binary extension field operations for arbitrary prime numbers, irreducible polynomials and precision. The proposed embedded cryptographic Processor architecture could achieve full cryptography algorithm flexibility, High hardware-utilization and High Performance.In particular ASIC solution generally leads to a higher throughput rate at a lower cost, but it is inflexible. To mitigate the gap between GPP & ASIC realizations, application-specific cryptographic processors have been proposed, each with its own instruction-set architecture, data path design and target applications.

Index Terms-Cryptographic-processor, reconfigurable architecture, finite field arithmetic (FFA),modular addition.

I. INTRODUCTION

FROM THE DAWN OF CIVILIZATION, to the highly networked societies that we live in today communication has always been an integral part of our existence. What started as simple sign-communication centuries ago has evolved into many forms of communication today the Internet being just one such example. Methods of communication today include

Radio communication,
Telephonic communication,
Network communication,
Mobile communication.

All these methods and means of communication have played an important role in our lives, but in the past few years, network communication, especially over the Internet, has emerged as one of the most powerful methods of communication with an overwhelming impact on our lives.Such rapid advances in communications technology have also given rise to security threats to individuals and organizations. In the last few years, various measures and services have been developed to counter these threats. All categories of such measures and services, however, have certain fundamental requirements, which include

Confidentiality, which is the process of keeping information private and secret so that only the intended recipient is able to understand the information.

Authentication, which is the process of providing proof of identity of the sender to the recipient, so that the recipient can be assured that the person sending the information is who and what he or she claims to be

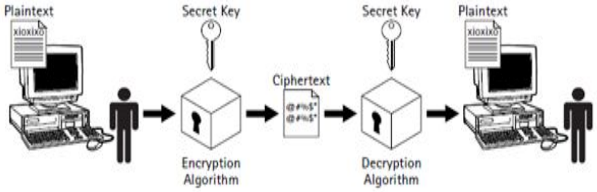


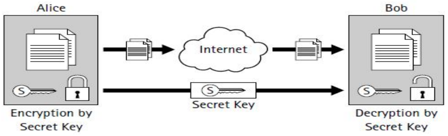
Figure 1: Conventional Encryption Model

Overview of Cryptography:
Let us now look at the various cryptography techniques available. For the purpose of classification, the techniques are categorized on the basis of the number of keys that are used. The two main cryptography techniques are

Single key cryptography: This cryptography technique is based on a single key. It is also known as symmetric key or private key or secret key encryption.

Public key cryptography: This cryptography technique is based on a combination of two keys—secret key and public key. It is also known as asymmetric encryption.Let us look at each of these methods in detail.

Single Key Cryptography:
The process of encryption and decryption of information by using a single key is known as secret key cryptography or symmetric key cryptography. In symmetric key cryptography, the same key is used to encrypt as well as decrypt the data. The main problem with symmetric key algorithms is that the sender and the receiver have to agree on a common key. A secure channel is also required between the sender and the receiver to exchange the secret key.



.

Figure 2: Secret key cryptography

Many secret key algorithms were developed on the basis of the concept of secret key cryptography. The most widely used secret key algorithms include

Data Encryption Standard (DES)
Triple-DES (3DES)
International Data Encryption Algorithm (IDEA)
RC4
RC5
CAST-12
RC6
Advanced Encryption Standard (AES).

Public Key Cryptography:

The approach called asymmetric cryptography evolved to address the security issues posed by symmetric Cryptography. This method solves the problem of secret key cryptography by using two keys instead of a single key. Asymmetric cryptography uses a pair of keys. In this process, one key is used for encryption, and the other key is used for decryption. This process is known as asymmetric cryptography because both the keys are required to complete the process. These two keys are collectively known as the key pair. In asymmetric cryptography, one of the keys is freely distributable. This key is called the public key and is used for encryption. The private key is not distributable. This key, like its name suggests, is private for every communicating entity. In public key cryptography, the data that is encrypted with the public key can only be decrypted with the corresponding private key. Conversely, data encrypted with the private key can only be decrypted with the corresponding public key. Due to this asymmetry, public key cryptography is known as asymmetric cryptography.

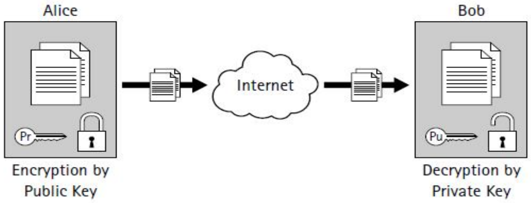


Figure 3: Public key encryption

This method very clearly indicates that the data you send to a user can only be encrypted by the public key. Similarly, the decryption can be done only by the private key, which is supplied by the recipient of the data. So, there is very little possibility of the data in transit being accessed or tampered by any other person. need to share a key, as required for symmetric encryption. All communications involve only public keys, and no private key is ever transmitted or shared. The above mechanism also brings out the point that every recipient will have a unique key that he will use to decrypt the data that has been encrypted by its counterpart public key. Diffie and Hellman first discussed the process of asymmetric cryptography.

Many Public key algorithms were developed on the basis of the concept of Public key cryptography. The most widely used Public key algorithms include

RSA.

ECC.

II ALGORITHMS OF SYMMETRIC CRYPTOGRAPHY:

DES:

DES is a block cipher: It encrypts/decrypts data in 64-bit blocks using a 64-bit key (although effective key length is 56-bit). DES is a symmetric algorithm: The same algorithm and key are used for both encryption and decryption. DES is an iterative cipher: the basic building block (a substitution followed by a permutation) called a round is repeated 16 times. For each DES round, a sub-key is derived from the original key called key schedule. Key schedule for encryption and decryption is the same except for the minor difference in the order (reverse) of the sub-keys for decryption.Encryption begins with an initial permutation (IP), which scrambles the 64-bit plain-text in a fixed pattern. The result of the initial permutation is sent to two 32-bit registers, called the right half register and left half register. Those registers hold the two halves of the intermediate results through succeeding 16 iterations. The contents of the right half register are permuted (permutation E) and sent to an exclusive-OR unit along with the sub-key for each iteration. Note that some bits are selected twice, allowing the 32-bit register to expand to 48 bits. The 48-bit output of the exclusive-OR block is divided into eight groups (6-bits each) to address eight substitution memories (S-boxes). A permutation *P* is applied to 32-bit output from S-boxes and then feed into an exclusive-OR block along with the contents of the left half register. The output of this block is written into temporary register, concluding the first iteration. At the next clock cycle, the contents of the temporary registers are written into the right half register and previous contents of the right half register are written into left half register. This process repeats through 16 iterations. After the 16 iterations,the right half and left half register contents are subjected to a final permutation IP¡1, which is the inverse of the initial permutation. The output of IP¡1 is the 64-bit cipher-text.

International Data Encryption Algorithm

International Data Encryption Algorithm (IDEA) is a block cipher; it operates on 64 bit plaintext blocks and uses 128-bit long input key K. The design philosophy behind this algorithm is “mixing operations from different algebraic group”. The 64-bit input block is divided into four 16-bit sub-blocks: X1, X2, X3, X4, which become the input blocks of the first round of the algorithm (figure 1). There are a total of 8 rounds. In each rounds, the four sub-blocks are XORed, Added, and multiplied with one another and with six 16 bit sub-blocks of key material. Between each round the second and the third blocks are swapped. Finally the four sub-blocks are combined with four sub-keys in an output transformation. The following are the basic operation used.

Bit-by-bit exclusive-OR of two 16-bit sub-blocks;

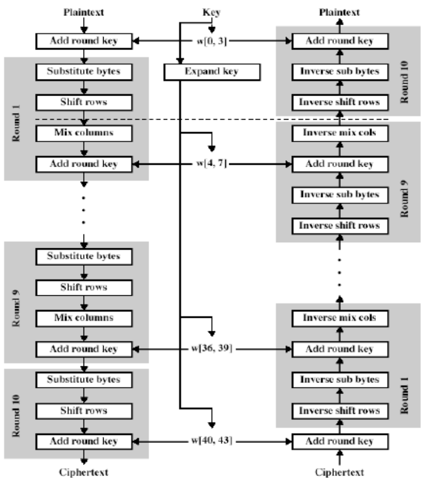
Addition of integers modulo 2^{16} where the 16-bit sub-block is treated as an unsigned integer;

Multiplication of integers modulo $2^{16} + 1$ where the 16-bit sub-block is treated as an unsigned integer except that all-zero sub-block is treated as representing 2^{16};

Key scheduling: The algorithm uses 52 sub-keys (six for each of eight rounds and four more for the output transformation). First, the 128-bit key is divided into eight 16-bit sub-keys. These are the first eight sub-keys for the algorithm (the six for the first round, and the first two for the second round). Then the key is rotated 25 bits to the left and again divided into eight sub-keys. The first four are used in round 2, the last four are used in round 3. The key is rotated another 25 bits to the left for the next eight sub-keys, and so on until the end of the algorithm The decryption scheme is the same as encryption’s scheme, except it utilizes a different set of sub keys generated from the key of IDEA where Ki denotes encryption sub keys and Ui denote decryption sub keys, where 1 < i < 52.

AES:

AES is a block cipher developed in effort to address threatened key size of Data Encryption Standard (DES).It allows the data length of 128, 192 and 256 bits, and supporting three different key lengths, 128, 192, and 256 bits. AES can be divided into four basic operation blocks where data are treated at either byte or bit level. The byte structure seems to be natural for low profile microprocessor (such as 8-bit CPU and microcontrollers) The array of bytes organized as a 4×4 matrix is also called "state" and those four basic steps; BytesSub,ShiftRow, Mix columns, and AddRoundKey are also known as layers. These four layer steps describe one round of the AES. The number of rounds is depended on the key length, i.e., 10, 12 and 14 rounds for the key length of 128, 192 and 256 bits respectively.The block diagram of the system with 128 bit data is shown below :



(a) Encryption (b) Decryption

Figure 4: AES Block Diagram

Substitute bytes Transformation : This operation is a non-linear byte substitution. It composes of two sub-transformations; multiplicative inverse and affine transformation. In most implementations, these two sub-steps are combined into a single table lookup called S-Box.

ShiftRow Transformation : This step is a simple permutation process, operates on individual rows, i.e. each row of the array is rotated by a certain number of byte positions.

Mix columns Transformation : This is a substitution step that makes use of arithmetic over GF (28). Column vector is multiplied (in GF (28)) by a fixed matrix where bytes are treated as a polynomial of degree less than 4.

AddRoundKey : Each byte of the array is added (respect to GF (2)) to a byte of the corresponding array of round subkeys. Excluding the first and the last round, the AES with 128 bit round key proceeds for nine iterations. Round keys are generated by a procedure called round key expansion or key scheduling. Those sub-keys are derived from the original key by XOR the two previous columns. For columns that are in multiples of four, the process involves round constants addition, S-Box and shift operations.

III OPERATIONS OF SYMMETRIC KEY ALGORITHM:

Modular Addition Two:

The addition of two elements in a finite field is achieved by “adding” the coefficients for the corresponding powers in the polynomials for the two elements. The addition is performed with the XOR operation (denoted by ⊕) i.e., modulo 2 -so that 1⊕1 = 0, 1 ⊕ 0 = 1 , and 0 ⊕ 0 = 0.

Alternatively, addition of finite field elements can be described as the modulo 2 addition of corresponding bits in the byte. For two bytes {a7a6a5a4a3a2a1a0} and {b7b6b5b4b3b2b1b0}, the sum is {c7c6c5c4c3c2c1c0}, where each ci = ai ⊕bi (i.e., c7 = a7 ⊕ b7, c6 = a6 ⊕ b6, ...). For example, the following expressions are equivalent to one another:

$$(x^6+x^4+x^2+x+1)+(x^7+x+1)=x^7+x^6+x^4+x^2 \quad \text{(polynomial notation);}$$

$$\{01010111\}\oplus\{10000011\}=\{11010100\} \quad \text{(binary notation);}$$

$$\{57\}\oplus\{83\}=\{d4\} \quad \text{(hexadecimal notation).}$$

Algorithm: For Modular addition Two.

```
Require: Binary Polynomials a (z), b (z) with maximum degree m-1.
Ensure: c(z)=a(z) + b(z).
1: for i from 0 to M-1 do
2: C[i] ← A[i] ⊕ B[i].
3: end for
4: Return(c).
```

Modular Multiplication 2^8:

In the polynomial representation, multiplication in GF (2^8) (denoted by •) corresponds with the multiplication of polynomials modulo an irreducible polynomial of degree 8. A polynomial is irreducible if its only divisors are one and itself. For the AES algorithm, this irreducible_polynomial is

$$m(x)=x^8+x^4+x^3+x+1,$$

For example, {57} • {83} = {c1}, because

$$\begin{aligned}(x^6+x^4+x^2+x+1)\,(x^7+x+1) &= x^{13}+x^{11}+x^9+x^8+x^7+\\ &\quad x^7+x^5+x^3+x^2+x+\\ &\quad x^6+x^4+x^2+x+1\\ &= x^{13}+x^{11}+x^9+x^8+x^6+x^5+x^4+x^3+1\end{aligned}$$

and

$$x^{13}+x^{11}+x^9+x^8+x^6+x^5+x^4+x^3+1 \text{ modulo } (x^8+x^4+x^3+x+1)$$

$$\begin{array}{r|l} & x^5+x^3 \\ \hline x^8+x^4+x^3+x+1 & x^{13}+x^{11}+x^9+x^8+x^6+x^5+x^4+x^3+1 \\ & x^{13}\qquad +x^9+x^8+x^6+x^5 \\ \hline & x^{11}\qquad\qquad\qquad +x^4+x^3 \\ & x^{11}\qquad +x^7+x^6\quad +x^4+x^3 \\ \hline & x^7+x^6\qquad\qquad +1 \end{array}$$

In Prime Field operations modulo means divide it requires more time,so in binary field operation it requires less time with simple addition.

$$x^{13}+x^{11}+x^9+x^8+x^6+x^5+x^4+x^3+1 \quad \oplus \ (x^8+x^4+x^3+x+1)$$

$$= \qquad x^7+x^6+1\,.$$

Matrix Multiplication:

Four - term polynomials can be defined - with coefficients that are finite field elements - as:

$$a(x)=a_3x^3+a_2x^2+a_1x+a_0 \qquad (1)$$

which will be denoted as a word in the form [*a*0, *a*1, *a*2, *a*3]. Note that the polynomials in this section behave somewhat differently than the polynomials used in the definition of finite field elements, even though both types of polynomials use the same indeterminate, *x*. The coefficients in this section are themselves finite field elements, i.e., bytes, instead of bits; also, the multiplication of four-term polynomials uses a different reduction polynomial, defined below. The distinction should always be clear from the context.

To illustrate the addition and multiplication operations, let

$$b(x)=b_3x^3+b_2x^2+b_1x+b_0 \ (2)$$

Define a second four-term polynomial. Addition is performed by adding the finite field coefficients of like powers of *x*. This addition corresponds to an XOR operation between the corresponding bytes in each of the words – in other words, the XOR of the complete word values.

Thus, using the equations of (1) and (2),

$$a(x)+b(x)=(a_3\oplus b_3)x^3+(a_2\oplus b_2)x^2+(a_1\oplus b_1)x+(a_0\oplus b_0) \quad (3)$$

Multiplication is achieved in two steps. In the first step, the polynomial product *c*(*x*) = *a*(*x*) *b*(*x*) is algebraically expanded, and like powers are collected

$$c(x)=c_6x^6+c_5x^5+c_4x^4+c_3x^3+c_2x^2+c_1x+c_0 \quad (4)$$

Where

$$\begin{array}{ll} c_0=a_0\bullet b_0 & c_4=a_3\bullet b_1\oplus a_2\bullet b_2\oplus a_1\bullet b_3 \\ c_1=a_1\bullet b_0\oplus a_0\bullet b_1 & c_5=a_3\bullet b_2\oplus a_2\bullet b_3 \\ c_2=a_2\bullet b_0\oplus a_1\bullet b_1\oplus a_0\bullet b_2 & c_6=a_3\bullet b_3 \end{array}$$

(5)

$$c_3=a_3\bullet b_0\oplus a_2\bullet b_1\oplus a_1\bullet b_2\oplus a_0\bullet b_3 .$$

The result, *c*(*x*), does not represent a four-byte word. Therefore, the second step of the multiplication is to reduce *c*(*x*) modulo a polynomial of degree 4; the result can be reduced to a polynomial of degree less than 4. For the AES algorithm, this is accomplished with the polynomial x^4 + 1, so that

$$x^i \bmod (x^4+1)=x^{i \bmod 4}.$$

The modular product of *a*(*x*) and *b*(*x*), denoted by *a*(*x*) ˜ *b*(*x*), is given by the four-term polynomial *d*(*x*), defined as follows:

$$d(x)=d_3x^3+d_2x^2+d_1x\ +d_0$$

with

$$\begin{aligned} d_0&=(a_0\bullet b_0)\oplus(a_3\bullet b_1)\oplus(a_2\bullet b_2)\oplus(a_1\bullet b_3) \\ d_1&=(a_1\bullet b_0)\oplus(a_0\bullet b_1)\oplus(a_3\bullet b_2)\oplus(a_2\bullet b_3) \\ d_2&=(a_2\bullet b_0)\oplus(a_1\bullet b_1)\oplus(a_0\bullet b_2)\oplus(a_3\bullet b_3) \\ d_3&=(a_3\bullet b_0)\oplus(a_2\bullet b_1)\oplus(a_1\bullet b_2)\oplus(a_0\bullet b_3) \end{aligned}$$

(6)

When *a*(*x*) is a fixed polynomial, the operation defined in equation (6) can be written in matrix form as:

$$\begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{bmatrix} = \begin{bmatrix} a_0 & a_3 & a_2 & a_1 \\ a_1 & a_0 & a_3 & a_2 \\ a_2 & a_1 & a_0 & a_3 \\ a_3 & a_2 & a_1 & a_0 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} \quad (7)$$

Because x^4 +1 is not an irreducible polynomial over GF(2^8), multiplication by a fixed four-term polynomial is not necessarily invertible.

Fixed Coefficient Multiplier.

Multiplying the binary polynomial defined in equation with the polynomial *x* results in

$$b_7x^8+b_6x^7+b_5x^6+b_4x^5+b_3x^4+b_2x^3+b_1x^2+b_0x .$$

The result x □□b(x) is obtained by reducing the above result modulo *m*(*x*), as defined in equation (4.1). If *b7* = 0, the result is already in reduced form. If *b*7 = 1, the reduction is accomplished by subtracting (i.e., XORing) the polynomial *m*(*x*). It follows that multiplication by *x* (i.e., {00000010} or {02}) can be implemented at the byte level as a left shift and a subsequent conditional bitwise XOR with {1b}. This operation on bytes is denoted by xtime(). Multiplication by higher powers of *x* can be implemented by repeated application of xtime(). By adding intermediate results, multiplication by any constant can be implemented.

$$\begin{aligned} \{57\}\bullet\{02\} &= \mathrm{xtime}(\{57\})=\{ae\} \\ \{57\}\bullet\{04\} &= \mathrm{xtime}(\{ae\})=\{47\} \\ \{57\}\bullet\{08\} &= \mathrm{xtime}(\{47\})=\{8e\} \\ \{57\}\bullet\{10\} &= \mathrm{xtime}(\{8e\})=\{07\}, \end{aligned}$$

thus,

$$\begin{aligned} \{57\}\bullet\{13\} &= \{57\}\bullet(\{01\}\oplus\{02\}\oplus\{10\}) \\ &= \{57\}\oplus\{ae\}\oplus\{07\} \\ &= \{fe\}. \end{aligned}$$

For example {57} • {13} = { fe } because

There are many ways to implement a finite field multiplier. An originally proposed one in the AES takes the form of Xtime() which is essentially multiplied by *x* or left-shift with {1B} feedback. That could imply either a bit-serial or a bit-parallel architecture. Rudra Proposed the implementation of Rijndael system with composite field arithmetic. We are considering a fast multiplier , simple , small area , and support pipeline architecture (if needed). Notice of the fix-value multiplications (by {02} or by {03}) leads us to a fixed-coefficient multiplication in GF(2^8) that fulfils our requirements. We are investigating this multiplier..

Let Si ,c = B*(x)* be an element to be multiplied. B*(x)* can also be written in the polynomial form as;

$$\begin{aligned} \mathrm{B}(x)=\mathrm{b}_0+\mathrm{b}_1x+\mathrm{b}_2x^2+\mathrm{b}_3x^3 \\ +\mathrm{b}_4x^4+\mathrm{b}_5x^5+\mathrm{b}_6x^6+\mathrm{b}_7x^7 \end{aligned}$$

Where b € (0,1).

Multiplications used in the Mix Column transformation are {03}.B(x)=(x+1)B(x) and

{02}.B(x) = x.B(x). The resulted multiplications are:

$$\begin{aligned} \{03\}\bullet\mathrm{B}(x)=(\mathrm{b}_0\oplus\mathrm{b}_7)+(\mathrm{b}_0\oplus\mathrm{b}_1)x \\ +(\mathrm{b}_1\oplus\mathrm{b}_2)x^2+(\mathrm{b}_2\oplus\mathrm{b}_3)x^3 \\ +(\mathrm{b}_3\oplus\mathrm{b}_4)x^4+(\mathrm{b}_4\oplus\mathrm{b}_5)x^5 \\ +(\mathrm{b}_5\oplus\mathrm{b}_6)x^6+(\mathrm{b}_6\oplus\mathrm{b}_7)x^7 \end{aligned}$$

$$\begin{aligned} \{02\}\bullet\mathrm{B}(x)=\mathrm{b}_7+(\mathrm{b}_0)x+\mathrm{b}_1x^2\ +(\mathrm{b}_2)x^3 \\ +(\mathrm{b}_3)x^4+\mathrm{b}_4x^5+\mathrm{b}_5x^6+\mathrm{b}_6x^7 \end{aligned}$$

Implementations of above equations are simple since additions are simply XORs. As an example the circuit to compute x.Bi is shown in Fig (3) below. The implementation of *(*x + 1*)* Bi shown in Fig (4). can be done similarly. According to terms given in (2), and an architecture shown in Fig.(4) , the maximum delay time is expected to be that of the a delay unit of a 2-input XOR gate.

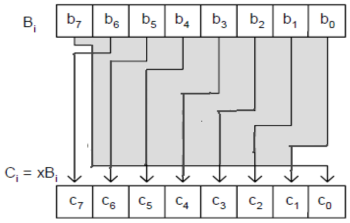


Figure 5:
A×2 Fixed Coefficient Multiplier

.

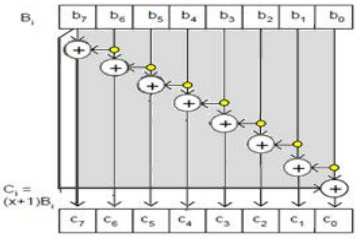


Figure 6: A×3 Fixed Coefficient Multiplier.

Mix Columns () Transformation:

The Mix column() Transformation operates on the state column-by-column as a four-term polynomial .The columns are considered as polynomials over GF(2^8) and multiplied modulo x^4 + 1 with a fixed polynomial *a*(*x*), given by

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

As described in Sec. 3.3, this can be written as a matrix multiplication. Let *s*¢(*x*) = *a*(*x*) ˜ *s*(*x*):

$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix} \quad \text{for } 0 \le c < Nb.$$

As a result of this multiplication, the four bytes in a column are replaced by the following

$$s'_{0,c} = (\{02\} \bullet s_{0,c}) \oplus (\{03\} \bullet s_{1,c}) \oplus s_{2,c} \oplus s_{3,c}$$

$$s'_{1,c} = s_{0,c} \oplus (\{02\} \bullet s_{1,c}) \oplus (\{03\} \bullet s_{2,c}) \oplus s_{3,c}$$

$$s'_{2,c} = s_{0,c} \oplus s_{1,c} \oplus (\{02\} \bullet s_{2,c}) \oplus (\{03\} \bullet s_{3,c})$$

$$s'_{3,c} = (\{03\} \bullet s_{0,c}) \oplus s_{1,c} \oplus s_{2,c} \oplus (\{02\} \bullet s_{3,c}).$$

By Using fixed coefficient multiplier we can implement the mix columns from equation 3.5.2 and 3.5.3 we can reduce the multiplication. state column by column matrix.

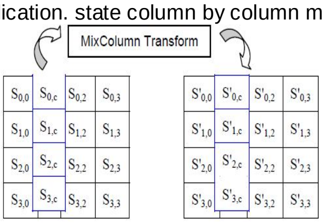


Figure 7: Mix Coloumn Transform.

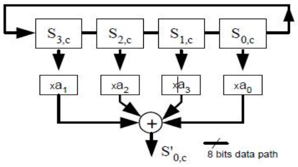


Figure8:MixColumn Transform Architecture

Multiplier X(2X+1) Modulo 2^8:

RC6 algorithm requires this operation from converting plain text to cipher text. Multiplying 2x refer in fixed coefficient multiplier. To reduce multiplication.

Let,

$X = b_0 + b_1x + b_2x^2 + b_3x^3 + b_4x^4 + b_5x^5 + b_6x^6 + b_7x^7$

Now ,

$\{02\}.B(x) + 1 = (b_7 + 1) + (b_0)x + b_1x^2 + b_2x^3 + b_3x^4 + b_4x^5 + b_5x^6 + b_6x^7$

$x.(\{02\}.x + 1\} \bmod 2^8 = x.(\{02\}.\{x\} + 1) \oplus x^8$

operation requires less time to implement RC6 Algorithm.

IV CRYPTOGRAPHIC PROCESSOR ARCHITECTURE

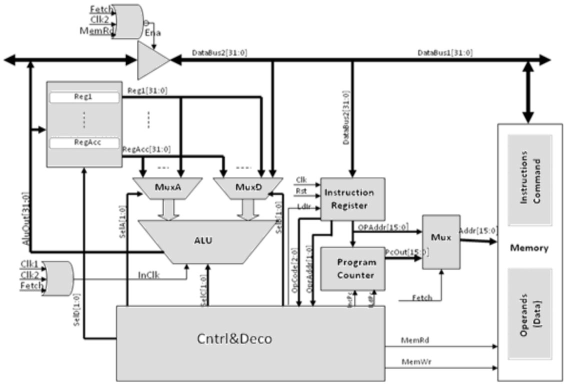


Figure 9: Cryptographic Processor

V IMPLEMENTATION

Modules Design of Architecture

Control and Decoder

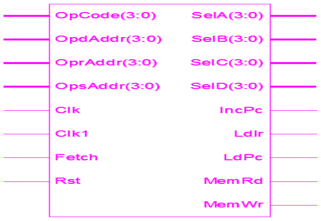


Figure 10: Block diagram of control and decoder

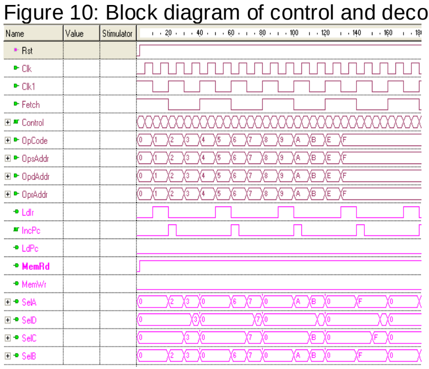


Figure 11: Simulation Results Of Control And Decode

General Purpose Registers.

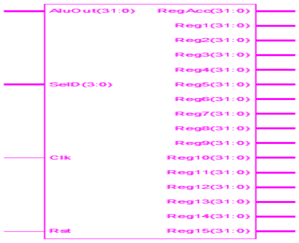


Figure 12: Block Diagram of General Purpose Register.

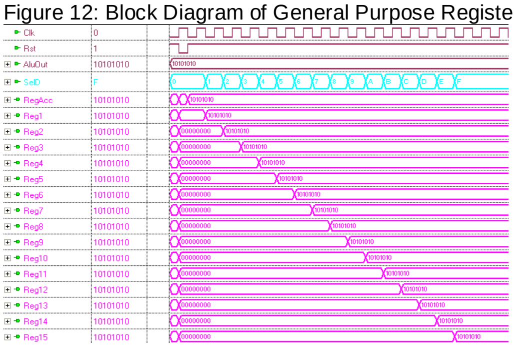


Figure 13 : Simulated Timing diagram of General Purpose Registers

Instruction Register

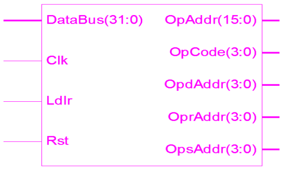


Figure 14 : Block Diagram Of Instruction register

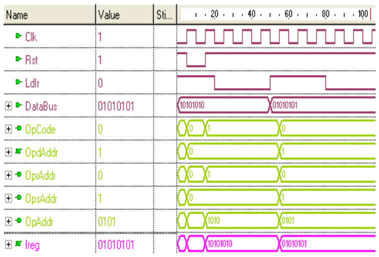


Figure 15: Simulated Timing diagram of Instruction Register
Program Counter

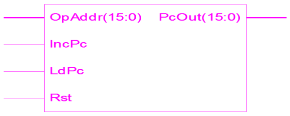


Figure 16: Block Diagram Of Program Counter.

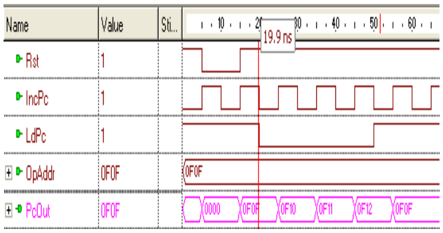


Figure 17:Simulated Timing diagram of Program counter

Mux(2:1):

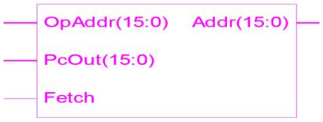


Figure 18: Block Diagram Of Mux(2:1)

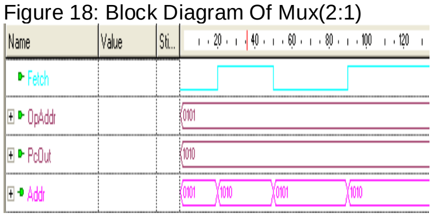


Figure 19: Simulated Timing diagram of Mux

MuxA(16:1)

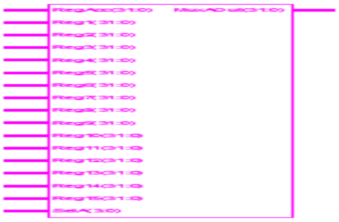


Figure 20: Block Diagram Of Mux(16:1).

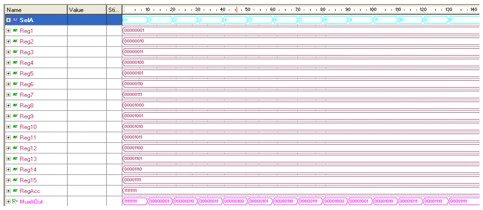


Figure 21:Simulated Timing diagram of MuxA(16:1)

Memory

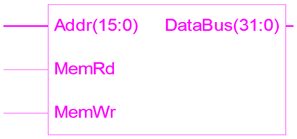


Figure 22: Block Diagram Of Memory

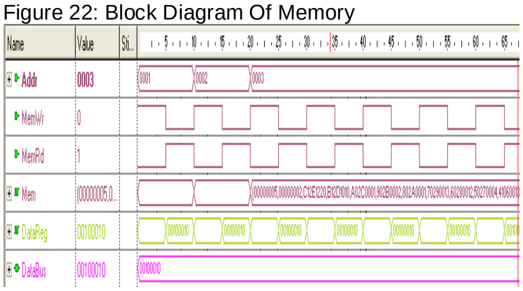


Figure 23 : Simulated Timing diagram of Memory
MuxD:

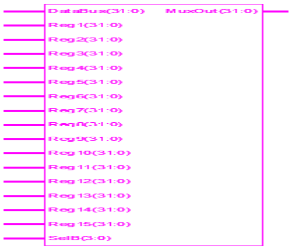


Figure 24: Block Diagram Of MuxD

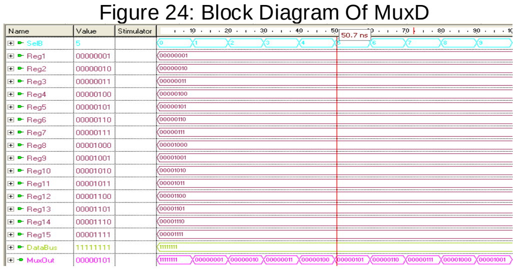


Figure 25: Simulated Timing diagram of MuxD

Arithmetic logical unit (ALU):

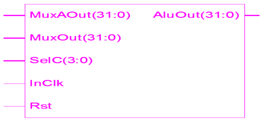


Figure 26 Block Diagram Of ALU

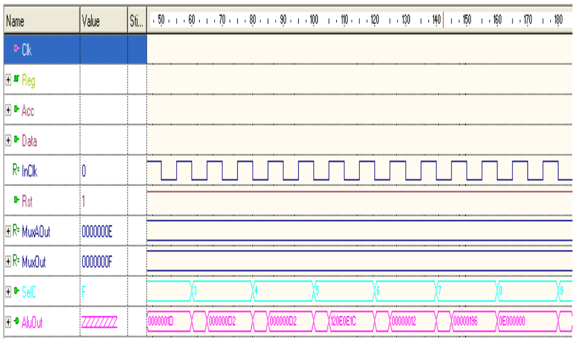


Figure 27Simulated Timing diagram of ALU

VI RESULT

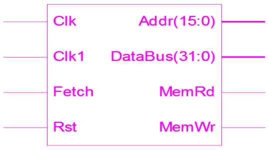
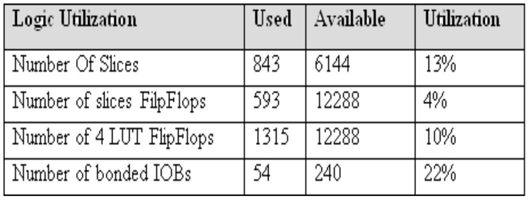


Figure 28: Block Diagram of Top Module

The Overall processor is designed in verilog and simulated using active HDL and synthesized on Xilinx virtex4.
After synthesizing the complete processor the number of slice flip flops utilization is minimal due to the combinational nature of the processor being capable of executing an instruction in few clock cycles .Maximum combinational path delay is 6.509ns

Table 1: Result



Logic Utilization	Used	Available	Utilization
Number Of Slices	843	6144	13%
Number of slices FilpFlops	593	12288	4%
Number of 4 LUT FlipFlops	1315	12288	10%
Number of bonded IOBs	54	240	22%

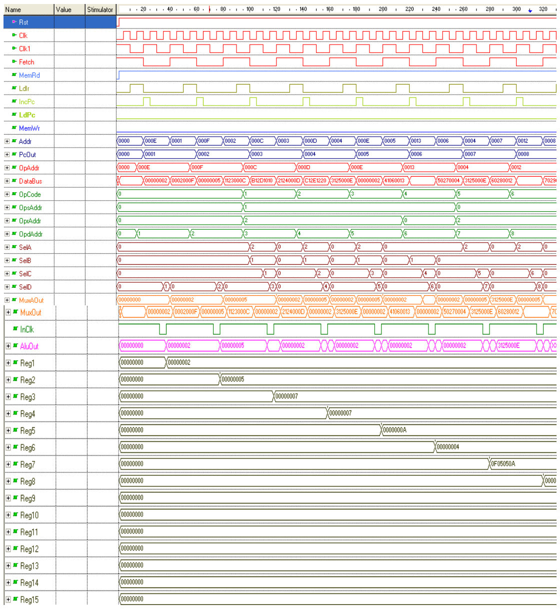


Figure 29: Simulated Timing diagram of Top Module

VII CONCLUSION

Thus the 32 bit cryptographic Processor perform mathematical computations used in Symmetric Key Algorithms has been designed using verilog The simulations are done with Active HDL simulator. The design is verified through exhaustive simulations. Thus processor architecture follows that one instruction executes in one clock cycle. The cryptographic processor concept proved that 20% of instruction did 80% of the work By this we increase overall performance of the speed with low area and low propagation delay.

REFERENCES

[1] Antonio H. Zavala “RISC Based Architecture for Computer Hardware Introduction Edición,, 2011 IEEE.

[2] NIST, "Advanced Encryption Standard (AES), (FIPPUB 197)", November 26, 2001, http://csrc.nist.gov/publications/.

[3] A. Rudra et. al., "Efficient Implementation of Rijndael Encryption with Composite Field Arithmetic", Proc.CHES2001, LNCS Vol. 2162, pp.175-188, 2001.

[4] Rohit Sharma, Vivek Kumar Sehgal, Nitin Nitin1, Pranav Bhasker, Ishita Verma , 2009, “Design And Implementation Of 64-Bit RISC Processor Using Computer Modeling And Simulation, pp. 568 – 573.

[5]R. Uma / International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622 www.ijera.com Vol. 2, Issue 2, Mar-Apr 2012, pp.053-058

[6]IEEE TRANSACTIONS on very large scale integration (VLSI) systems, vol. 18, No 8, August 2010 1145 A High-Performance Unified-Field Reconfigurable Cryptographic Processor Jun-Hong Chen, Ming-Der Shieh, Member, IEEE, and Wen-Ching Lin.

[7]FPGA Implementations of the RC6 Block Cipher Jean-Luc Beuchat Laboratoire de l’Informatique du arall´elisme, Ecole

Normale Sup´erieure de Lyon,46, All´ee d'Italie, F–69364 Lyon Cedex 07,Jean-Luc.Beuchat@ens-lyon.fr.

[8] Some Guidelines for Implementing Symmetric-Key Cryptosystems on Recon¯gurable-Hardware Arturo Daz-Perez, Nazar A. Saqib, and Francisco Rodrguez-Henriquez Computer Science Section, Electrical Engineering Department Centro de Investigaci¶on y de Estudios Avanzados del IPN Av. Instituto Politecnico Nacional No. 2508, Mexico D.F.fnabbas@ computacion.cs.cinvestav.mx, adiaz, Francisco @cs. cinvestav.mxg.

[9]Imyong lee, Dongwook Lee, Kiyoung choi "ODALRISC: A Small, Low power and Configurable 32-bit RISC processor" International SOC design conference 2008.

[10].Wa yne Wolf, FPGA Based System Design , Prentice Hall, 2005.

[11] R. Razdan and M.D. Smith, "A High-Performance Micro architecture with Hardware-Programmable Functional Units,"Proc. Micro-27, IEEE Computer Society, 1994, pp. 172-180.

[12].Vincen tP. Heuring, and Harry F. Jordan, "Computer Systems Design and Architecture " , 2n d Edition, 2003.

[13] The Practical XILINX Designer Lab Book, Dave Van den Bout, ISBN 0-13- 095502-7, p 30-31.

[14] XILINX datasheet library, http:// www.xilinx.com/ partinfo/4000.pdf

[15] A 32-b RISC/DSP microprocessor with reduced complexity Dolle, M. Jhand, S. Lehner, W. Muller, O. Schlett, M. Hyperstone Electron., Konstanz Date of Current Version: 06 August 2002

[16] VHDL-based development of a 32-b pipelined RISC processor for educational Purposes Buhler, M. Baitinger, U.G. Stuttgart Univ. Date of Current Version: 06 August 2002

AUTHORS

First Author- Manisha.Guduri, M.Tech -VLSI System Design,Assistant Professor in Alamuri Ratnamala Institute of Engineering & Technology.
guduri.manisha8@gmail.com

Second Author- V Jaya Rajesh, M.Tech-Power Systems, Assistant Professor in Alamuri Ratnamala In9stitute of Engineering & Technology. raj.raybon@gmail.com