M. Lalitha Sowmya, B.Divya, S.Jagadeesh / International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622 www.ijera.com
Vol. 2, Issue 5, September- October 2012, pp.1718-1724

Design of Custom Instructions in Cryptographic Processor

M. Lalitha Sowmya[1], B.Divya[2], S.Jagadeesh[3]
(Assistant Professor, Department of ECE, SSJ Engineering College, JNTU , Gandipet.)
(MTech Student, Department of ECE, SSJ Engineering College, JNTU, Gandipet.)
(Associate Professor and H.O.D, Department of ECE, SSJ Engineering College, JNTU, Gandipet.)

Abstract

In this paper, we are implementing 32 bit pipelined processor on FPGA and designed using verilog. In this Processor , we had performed logical operations and arithmetic operations like rotate word, modular addition modular multiplication, matrix multiplication, fixed coefficient multiplier ,mix column transform using binary extension field operations (2^m) for arbitrary irreducible Polynomial. Using our proposed field arithmetic units we can implement Symmetric Key Cryptography algorithms . Experimental Results Shows that developed processor working with high Speed , low area and low path delay .

Keywords: Cryptographic Processor, Pipeline, Finite field arithmetic (FFA), Symmetric Key Cryptography algorithms.

1. Introduction

The explosive growth in data communications and Internet services have made cryptography an important research topic. Cryptography is used for confidentiality, authentication, data integrity, and non-repudiation, which can be divided into two families:

Asymmetric key cryptography**:** In public key cryptography, the data that is encrypted with the public key can only be decrypted with the corresponding private key.

Symmetric key cryptography**:** The process of encryption and decryption of information by using a single key is known as Symmetric Key Cryptography. These are based on a mathematical function to encrypt a plain-text message and to produce cipher message. [8]

In this Paper we are designing Symmetric key mathematical operations in a 32 bit pipelined processor.

Implementing Symmetric Key operations in software seems to not only too slow for fast application such as Routers but also vulnerable to attacks. In contrast, in Hardware implementation, the higher data rate (G bits/second) is made possible by parallel and/or pipelining processing. Moreover, the implementations are physically

Secure since tempering by an outside attacker is Difficult. With these supporting reasons we are looking at the hardware implementation.

Implementing various symmetric-Key operations in a general-purpose Processor (GPP) is flexible but requires a lower throughput rate, more clock cycles for each instruction, more no. of addressing modes and larger power Consumption.

So we developed processor that the instruction set can be hardwired to speed instruction execution. No microcode is needed for single cycle execution. All instructions are one word (fixed bit) in length. This simplifies the instruction fetch mechanism since the location of instruction boundaries is not a function of the instruction type. The processor has small number of addressing modes. Only load and store instructions access memory. There are no computational instructions that access memory; load/store instructions operate between memory and a register. Control hardware is simplified and the machine cycle time is minimized.

The remainder of this paper is organized as follows. Section 2: Algorithms of Symmetric Key operations. Section 3: Implementation of Operations. Section 4 Proposed Architecture. Section 5: Modules design of ALU, Control unit, Multiplexers and general purpose registers. Section 6. Results. Section7**:** Conclusion Section 8**:** References.

2. Cryptographic algorithms for symmetric block ciphers.

Advanced Encryption Standard (AES), RC6, RC5, Data Encryption Standard (DES), Blowfish, International Data Encryption Algorithm (IDEA).

Blowfish is a symmetric block cipher that encrypts data in 8-byte (64-bit) blocks [3]. The algorithm has two parts, key expansion and data encryption. Key expansion consists of generating the initial contents of one array

Namely, eighteen 32-bit sub-keys, and four arrays (the S-boxes), each of size 256 by 32 bits, from a key of at most 448 bits (56 bytes). The data encryption uses a 16-round Feistel Network .The F Function, regarded as the

Primary source of algorithm security [3], combines two simple functions: **Addition modulo two** (XOR) and **Addition modulo** 2^32.

AES [2] 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 array of bytes organized as a 4×4 matrix is also called "state" and those four basic steps;**,** Bytes Sub **Shift Row or Rotate Word, Matrix Multiplication, Mix Column**, 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.

RC5 is exactly designated as RC5-w/r/b, where the variable parameters w, r, and b respectively denote the Word size (in bits), the number of rounds, and the length of secret key (in bytes). The allowable value of w is 16, 32 and 64; the allowable values of r and b range from 0 to 255. The parameter of RC5-32/12/16 is commonly chosen there are three routines in RC5: key expansion, encryption, and decryption. These routines consist of three primitive operations (and their inverse): **words addition , bitwise XOR, and data-dependent left rotation of x by y denoted by** x <<< y. Note that only the log2(w) low order bits of y affect this rotation. In the key-expansion routine, the user provided secret key is expanded to fill a key table whose size depends on the number of rounds. The key table is then used in both encryption and decryption.

RC6 **[**7] **is** a symmetric-key algorithm which encrypts 128-bit plaintext blocks to 128-bit cipher text blocks. The encryption process involves four operations: **Integer addition modulo** 2^w**, Bitwise exclusive or of two *w*-bit words,. Rotation to the left, and**

f(*X*) = (*X*(2*X* + 1)) mod 2^w **.**

IDEA [8] algorithm of the encryption process, we provide the original (128bits) cipher key to the mentioned unit. When necessary, the Key Generator Unit produces different sub-keys by performing **circular left shift operation (by 25bits)** on the current key and provides the sub-keys to other units. The unit named as "Multiplication modulo 216 + 1", is used to perform all the **multiplication modulo 2^16+1 operation, when required. The same is for unit "Addition modulo 2^16" and unit "Bitwise XOR".** or the parallel implementation of IDEA algorithm, the entire encryption process can be performed in several steps and performing operations in parallel wherever possible. Parallelism in operations can be achieved both in software and using hardware.

3. Implementation of Algorithm operations.

3.1 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 $\oplus$) i.e., modulo 2 -so that $1 \oplus 1 = 0$, $1 \oplus 0 = 1$, and $0 \oplus 0 = 0$.

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] $\oplus$ B[i].
3: end for
4: Return(c).

3.2 .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
$m(x) = x^{8.}$

. For Example,{57}.{83}={C1},because

$(x^6 + x + x^2 + x + 1)\ (x^7 + x + 1) =$

$x^{13} + x^{11} + x^9 + x^8 + x^7 + x^7 + x^5 + x^3 + x^2 + x + x^6 + x^4 + x^2 + x + 1 =$
$x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1.$
And
$x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1$ modulo $x^{8.}$.
In Prime Field operations modulo means divide it requires more time .so in binary field operation it requires less time with simple addition.
$x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1 \oplus x^{8.}$.
$= x^6 + x^5 + x^4 + x^3 + 1.$
{101011011110} $\oplus$ {0000010000000} = {111101}.

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

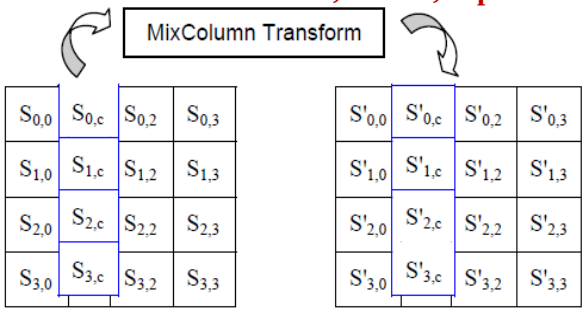


Fig 1: Mix Column Transform.

To derive a suitable Mix Column transform architecture, the trans formation matrix given in Fig can be rewritten as $s'(x) = s(x) \bullet a(x) \bmod (x^4 + 1)$, where • denotes finite field polynomial multiplication i.e.,

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

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

$S^1_{0,c} = (\{02\} \bullet S_{o,c}) \oplus (\{03\} \bullet S_{1,c}) \oplus S_{2,c} \oplus S_{3,c}$
$S^1_{1,c} = S_{o,c} \oplus (\{02\} \bullet S_{1,c}) \oplus (\{03\} \bullet S_{2,c}) \oplus S_{3,c}$
$S^1_{2,c} = \bullet S_{o,c} \oplus S_{1,c} \oplus (\{02\} \bullet S_{2,c}) \oplus (\{03\} \bullet S_{3,c})$
$S^1_{3,c} = (\{03\} \bullet S_{o,c}) \oplus \bullet S_{1,c} \oplus S_{2,c} \oplus (\{02\} \bullet S_{3,c})$

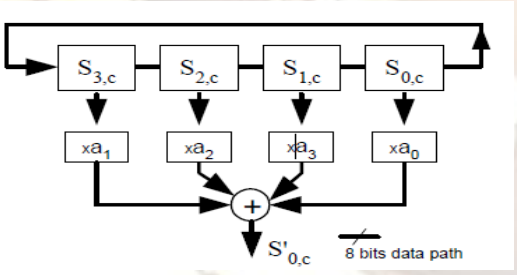


Fig .2 Mix Columns Transform Architecture

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 [3] 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...

3.5. Fixed Coefficients Multiplier.

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

$$B(x) = b_0 + b_1 x + b_2 x^2 + b_3 x^3 + b_4 x^4 + b_5 x^5 + b_6 x^6 + b_7 x^7 \quad \text{(Eq 3.5.1)}$$

Where $b \in (0,1)$.

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:

$\{03\}.B(x) = (b_0 \oplus b_7) + (b_0 \oplus b_1)x + (b_1 \oplus b_2) x^2 + (b_2 \oplus b_3)x^3 + (b_3 \oplus b_4) x^4 + (b_4 \oplus b_5) x^5 + (b_5 \oplus b_6) x^6 + (b_6 \oplus b_7) x^7$. (Eq: 3.5.2)

$\{02\}.B(x) = b_7 + (b_0) x + b_1 x^2 + b_2 x^3 + b_3 x^4 + b_4 x^5 + b_5 x^6 + b_6 x^7$ (Eq 3.5.3)

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.

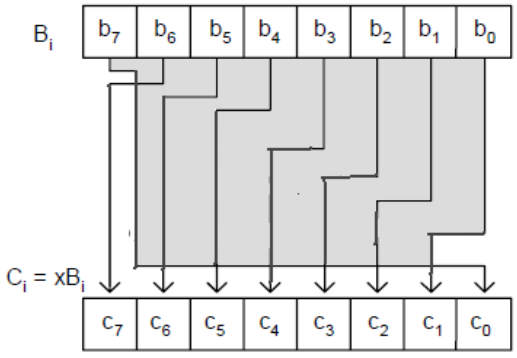


Fig 3: A×2 Fixed Coefficient Multiplier.

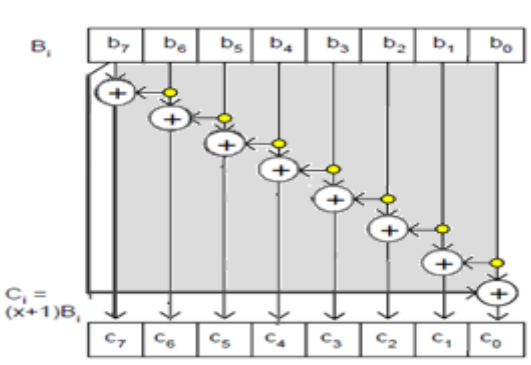


Fig *4:* A×3 Fixed Coefficient Multiplier.

3.6 Multiplier X (2X + 1) Modulo 2^8.

Let,

$$X = b_0 + b_1x + b_2x^2 + b_3x^3 + b_4x^4 + b_5x^5 + b_6x^6 + b_7x^7. \quad \text{(Eq 3.6.1)}$$

Now,

$$\{02\}.B(x) + 1 = (b_7 + 1) + (b_0)\,x + b_1x^2 + b_2x^3 + b_3x^4 + b_4x^5 + b_5x^6 + b_6x^7 \quad \text{(Eq 3.6.2)}$$

$$x.(\{02\}.x + 1) \bmod 2^8 = x.(\{02\}.\{x\} + 1) \oplus x^8 \quad \text{(Eq 3.6.3)}$$

Eq (3.6.3), operation requires less time to implement Rc6 Algorithm.

3.7 Shift Row Transform.

In the **Shift Rows()** transformation, the bytes in the last three rows of the State are cyclically shifted over
Different numbers of bytes.

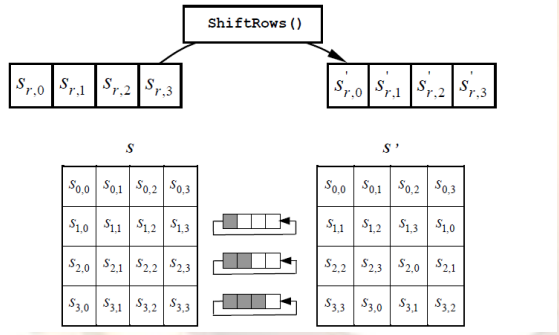


Fig 5. Shift Rows Architecture.

4. Cryptography Processor.

The architecture of an 32 bit processor is shown in Fig 6. The processor [1] is designed with load/store architecture. Separate memory for instructions (program) and data Different stages of the pipeline perform simultaneous Accesses to memory. This Harvard style of architecture can Either be used with two completely different memory Spaces, a single dual-port memory space with separate data and instruction.. Three stages of pipelining have been incorporated in the design which increases the speed of operation.
The processor presented instruction set and uses a Single Instruction – Single Data (SISD) execution order. Its main characteristics are:
• Sixteen 32-bit general purpose registers.
•ALU with basic arithmetic and logical operations.

In this processor we are performing various operations of cryptography so we called as cryptography processor.

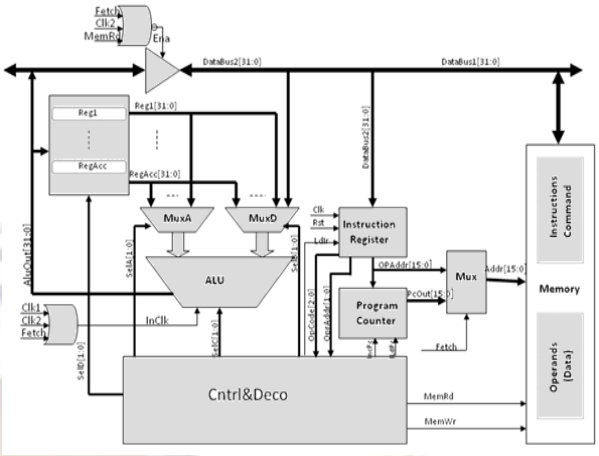


Fig 6. Cryptography Processor Architecture.

4.1 Instruction Set.

For a complete design, it was necessary to create a specific instruction set and its own assembly code with its proper instruction format.
The Instructions are classified into two groups.

•Data Manipulation (Load and Storage).
• Operations (Arithmetic and Logical).

The Logical operations like Shift Left, Shift Right, and Rotate Word Which requires only one Source Register. Shown in Type 3.
The Arithmetic Operations like addition ,modular functions ,etc to execute these operations we requires two source registers and to tore result in destination register. Shown in Type 2.
The Load instructions and store instructions Requires address from different data sources shown in Type 1.

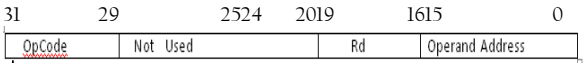
Table 1 describes complete Instruction set. Each Instruction having its own Opcode.As the complete set contains 13 instructions; 4 bits are enough to represent them.

Table:1 Instruction Set Of The Developed Processor.

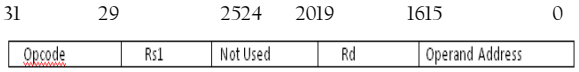
Syntax	operation	Description
Nop	nop	No operation.
Ld Sr,[A]	Sr = Mem[Addr].	Move data from memory to register sr.
Addition (A,B).	C = A ⊕ B.	GF(2^m) Addition.
ModularAddition (A,B).	C = A+B Mod P.	GF(2^m) Modular Addition.
Modular Multiplication (A,B).	C = A*B Mod P.	GF(2^m) Modular Multiplication.
Matrix Multiplication (A,B).	C = A*B Mod x4+1.	Polynomial GF(2^m)Matrix Multiplication.
MixColumn (A,B).	C = Y*A Mod x4+1.	Polynomial GF(2^m)Mix Column Transform.
FixedMultiplier (A).	C = {03}*A.	Reduction Multiplication.
AMXModulo (A).	C = A*(2A+1) Mod P.	Reduction Modulo Multiplication.
Length Rotation (A,B).	C = A<<B.	Variable length Rotation.
Rotate word (A).	C = ShiftRow(A).	. Rotate Word.
Left Shift (A)	C = A>>1.A<<1	Logical Left Shift,Right Shift.

C= Result . A,B = Operands . Y=Constant . sr =Source Register.[A]=Address.

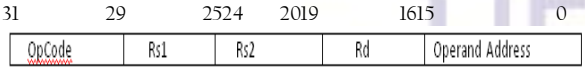
Type1.



Type2.



Type3.



5. Modules Design of Architecture.

5.1 Control Unit.

The control unit design is based on using FSM (Finite State Machine) and we designed it in a way that allows each state to run at one clock cycle, the first state is the reset which is initializes the CPU internal registers and variables. The machine goes to the reset state by enabling the reset signal for a certain number of clocks. Following the reset state Would be the instruction fetching and decoding states which will enable the appropriate signals for reading instruction data from the ROM then decoding the parts of the instruction. The decoding state will also select the next state depending on the instruction, since every instruction has its own set of states, the control unit will jump to the correct state based on the instruction given. After all states of a running instruction are finished, the last one will return to the fetch state which will allow us to process the next instruction in the program. Fig7: shows the state diagram for the control unit.

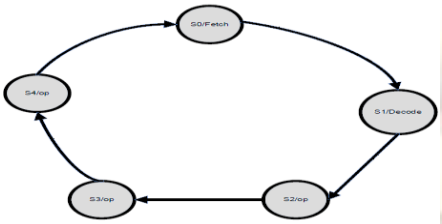


Fig 7. State Diagram of Control Unit.

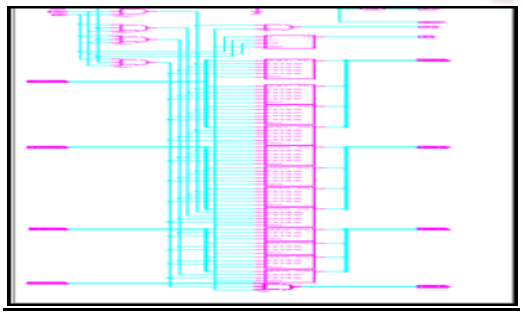


Fig 8: Top Block of Control and Decode.

5.2 General Purpose Registers.

General Purpose Registers (GPRs) store and save operands And results during program execution. ALU and memories must be able to write/read those registers, so a set of Sixteen 32-bit registers were used, along with multiplexers and control& decoder which register is read or written. These two registers are the Operands to ALU which performs the operation.

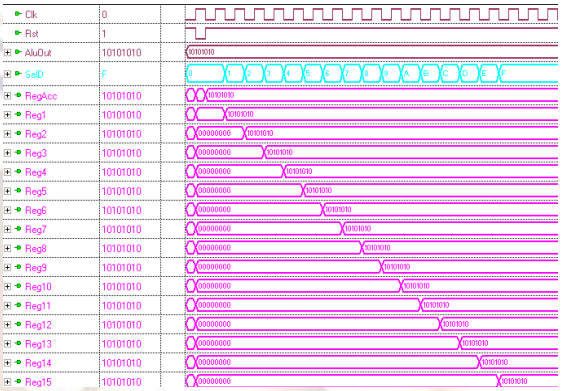


Fig 9: Simulated Timing diagram of General Purpose Registers.

5.3 Instruction Register.

Instruction registers store the instruction which read from the program memory, and keep it as an output for the decoder, which separates the operation code, Source Registers, Operand address and operands and these values will set to General purpose registers, Multiplexers and ALU to execute the command. This is achieved simply using buffers to translate data to/from the processor.

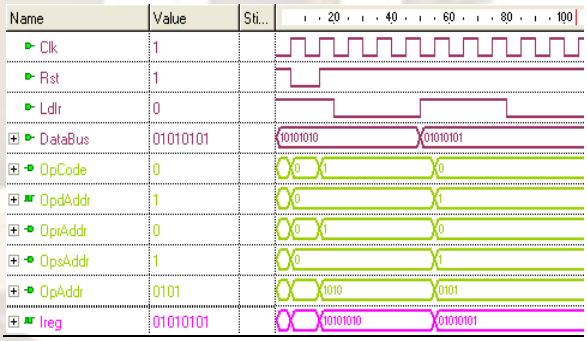


Fig 10: Simulated Timing diagram of Instruction Register

5.4 Arithmetic logical unit (ALU).

The Arithmetic-Logic Unit has 12 operations; each one of them was created and converted into a symbol, then, a multiplexor was placed in order to obtain a 4 bit selector

The ALU design comprises of 2 units. One unit is meant for logic operation and the other unit is meant f or Arithmetic operations shown in Table .1.

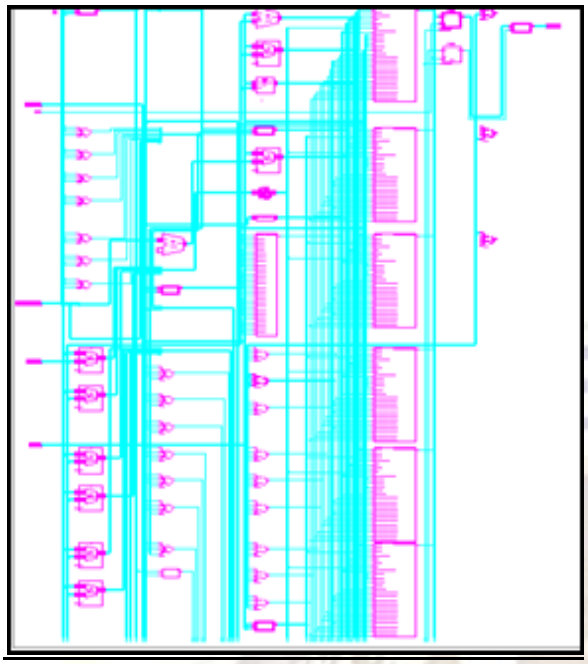


Fig 11: Top Block of ALU

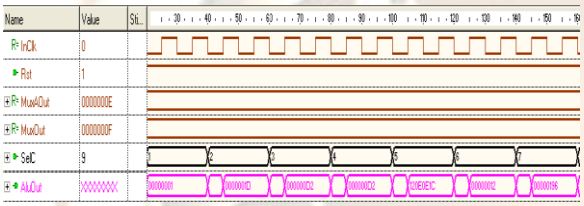


Fig 12: Simulated Timing diagram of ALU

.

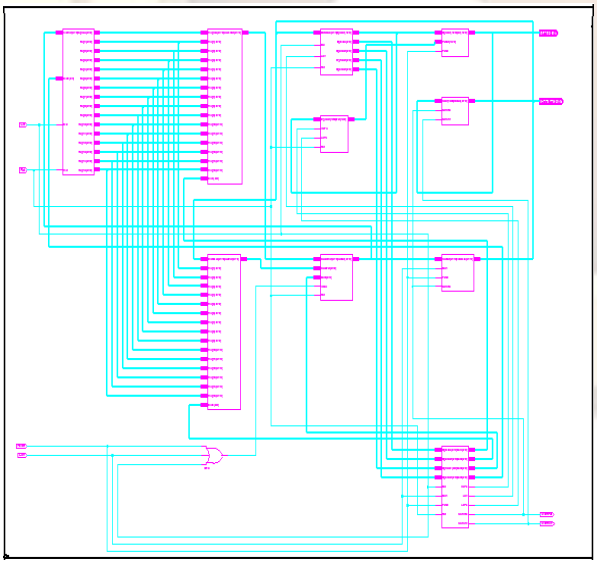


Fig 13: Top Block of 32 Bit Processor.

6. Results:

The ISE of the 32 bit processor was described using the Verilog .The tool chain including the Active HDL simulator and synthesized with the Xilinx 9.2i tool;

After synthesized the Hardware resource consumption for the complete processor implemented in a **Xilinx** Virtex4 **XC4VlX15-12Sf363** FPGA is shown in Table 2, 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.

Table 2: Hardware Resource Consumed

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%

For complete processor the total equivalent gate count for the complete processor is **14,518 gates ,** Maximum combinational path delay is **6.509ns** Maximum Frequency : **92.659MHz** , the area utilized only **13%.**

7. 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. By this we increase overall performance of the speed with low area and low propagation delay. In order to obtain a more sophisticated architecture is necessary to add some advanced techniques pipelining this processor can also perform floating point operations. And differential equations. Apart from this it can be used in portable gaming kits, Smart cards, ATMs.

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 Design and Performance Analysis of 8-bit RISC Processor using Xilinx Tool

[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 Reconfigurable-Hardware* Arturo ³az-P¶erez, Nazar A. Saqib, and Francisco Rodrguez-Henriquez Computer Science Section, Electrical Engineering Department Centro de Investigacion y de Estudios Avanzados del IPN Av. Instituto Politecnico Nacional No. 2508, Mexico D.F.*f*nabbas@ computacion.cs.cinvestav.mx, adiaz, Francisco @cs. cinvestav.mx*g*.

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

[10]. Wayne 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 t P. Heuring, and Ha rry F. Jordan, "*Computer Systems Design and Architecture*", 2n d E dition, 2003.