International Journal of Engineering Science Invention
ISSN (Online): 2319 – 6734, ISSN (Print): 2319 – 6726
www.ijesi.org Volume 2 Issue 2 ǁ February. 2013 ǁ PP.01-08

RISC Based Architecture for Customized Cryptographic Instructions

[1]Fathima Shireen, [2]Syed Musthak Ahmed,[3]G. Krishnamurthy

[1]MTech Student, S.R.Engineering College, Warangal
[2]Professor and H.o.D, ECE Dept, S.R.Engineering College, Warangal
[3]Professor, ECE Dept, S.R.Engineering College, Warangal

ABSTRACT: *Security is one of the most important features in data communication. Cryptographic algorithms are mainly used for this purpose to obtain confidentiality and integrity of data in communication. Implementing a Cryptographic algorithm on a general purpose processor results in lower throughput and larger power consumption. In this work, we propose processor architecture to perform the cryptographic algorithms that also speeds up the encryption and decryption process of data. This processor will perform the cryptographic operations like general instructions in GPP. The processor architecture is designed using Verilog HDL, with the data size of the processor of 32 bits.*

Keywords: Cryptographic Algorithms, GPP, Verilog, ALU, RISC, CISC.

I. INTRODUCTION

There are two basic types of processor design philosophies: reduced instruction set computer (RISC) and complex instruction set computer (CISC). As the name suggests CISC systems use complex instructions. For example adding two integers is considered a simple instruction. But an instruction that copies an element from one array to another and automatically updates both array subscripts is considered a complex instruction. RISC systems use only simple instructions. RISC systems assume that the required operands are in the processors internal registers not in the main memory. A CISC design does not impose such restrictions. RISC designs use hardware to directly execute instructions [3][8].

Cryptography plays a significantly important role in the security of data transmission. On one hand with developing computing technology implementation of sophisticated cryptographic algorithms has become feasible. The cryptographic algorithms are classified into public key cryptography and private key cryptography. The private key cryptography which usually has a relatively compact architecture and smaller key size than public key cryptography is often used to encrypt/decrypt sensitive information or documents. Some well known examples of public key cryptographic algorithms are RSA (Rivest-Shamir-Adleman) and elliptic curve crypto systems and private key cryptographic algorithms are AES (Advance Encryption Standard), DES (Data Encryption Standard) and TEA (Tinny Encryption Algorithm). Implementation of these cryptographic algorithms on a general purpose processor is complex and also it has the drawback of lower throughput and higher power consumption [1][9].

In the present work the design of a 32-bit data width RISC processor is presented based on cryptographic algorithms. It was designed with simplicity and efficiency in mind. It has a complete instruction set, Hayward architecture memory, general purpose registers and simple Arithmetical Logic Unit (ALU)[2]. Here the ALU design performs the cryptographic operations like operations in AES, Blowfish, and IDEA algorithms. To design of RISC architecture we used Verilog HDL.

Present work is divided as follows: Section II presents the Processor architecture with cryptographic operations; section III presents the Cryptographic operations .Section IV is dedicated with functional blocks and results.

The proposed processor has 32-bit data size, that its architecture has been designed in a way to be modular.

The ALU unit that uses a minimal instruction set, emphasizing the instructions used most often and optimizing them for the fastest possible execution. In this architecture the execution time of all instructions with the CPU clock cycle. The proposed architecture will perform both basic arithmetic and logical operations and cryptographic operations like Rotate word, Swapping, Fixed coefficient multiplication, matrix multiplication [3][6].

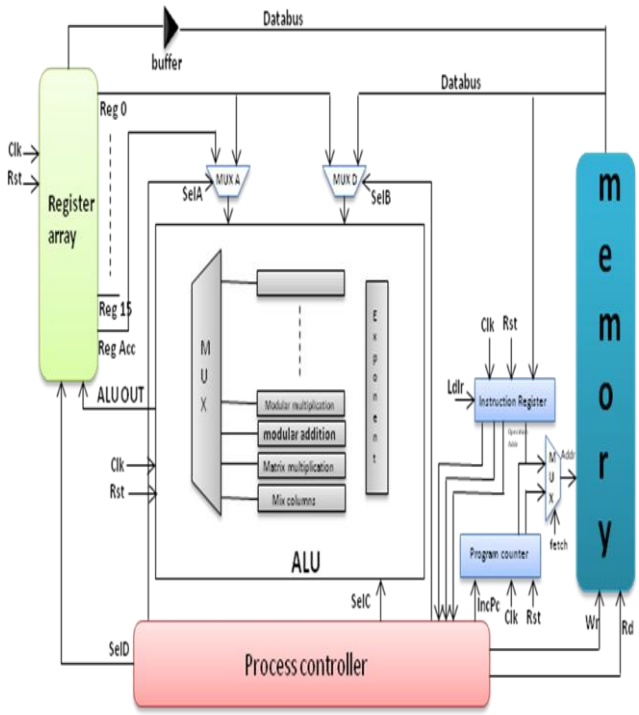


Fig.1: Processor Architecture

III. CRYPTOGRAPHIC OPERATIONS

AES (Advance Encryption Standard) is a block cipher developed in effort to address threatened key size of Data Encryption Standard (DES)[4]. It allows the data length of 128 bits and different key lengths 128, 192, 256 bits. The main operations in AES are Shift Rows, Rotate Word, Matrix Multiplication, Mix column [5].

Blowfish is a symmetric block cipher that encrypts data in 8-byte blocks. 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 (S-Boxes), each of size 256 by 32 bits from a key of at most 448 bits. The main operations of this algorithm are addition modulo two (XoR) and addition modulo 2^32.

IDEA algorithm of the encryption process we provide the original (128 bits) cipher key to the mentioned unit. When the necessary the key generator unit produces different sub-keys by performing circular left shift operation by 25 bits 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 unit is for bit wise Xor.

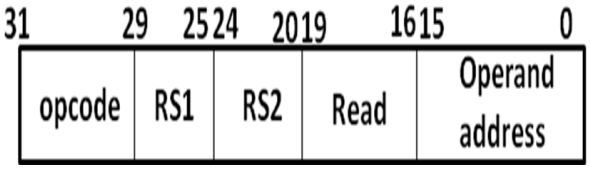
Instruction Set: for a complete design it was necessary to create a specific instruction set and its own instruction format. The instructions are classified in to Data manipulation and arithmetic logical operations.

The below table describes the complete instruction set. Each instruction having its own opcode.

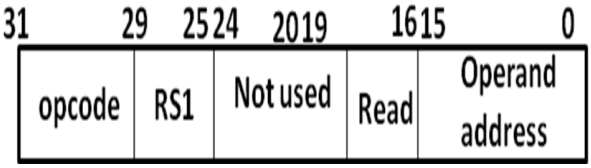
Table: Instruction Set

Syntax	Operation	Description
NoP	**Nop**	**No operation**
Ld Sr[A]	**Sr= Memm[Address]**	**Move data from memory to register**
Addition [A,B]	**C=A xor B**	**GF(2m) addition**
Modular Multiplication[A,B]	**C=A+Bmod P**	**GF(2m) modular addition**
Modular Multiplication[A,B]	**C=A*Bmod P**	**GF(2m) modular multiplication**
MatrixMultiplication[A,B]	**Matrix multiplication**	**Polynomial matrix multiplication**
Mix column[A,B]	**C=Y*A mod X 4%1**	**Polynomial mix column transformation**
Fixedmultiplier[A,B]	**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**
LRShift[A,B]	**C=A>>B,C=A<<B**	**Left, rotate shift operation**

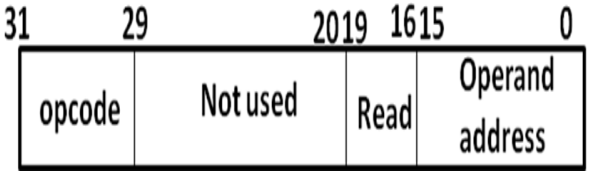
The logical operations like shift left, shift right and rotate word which requires only one source register shown in below type.



The operations like addition, modular functions require two source registers and to store result in destination result as shown in below type.



The load instructions and store instructions requires address from different data sources shown in below.



IV. RESULT

Instruction Register: Instruction registers store the instruction which read from the memory and keep it as an output for the control circuit like operation code, source registers, operand address and operands these values set to general purpose registers. Below shows the block diagram of instruction register, simulation results and technology schematic and a table of implementation results.

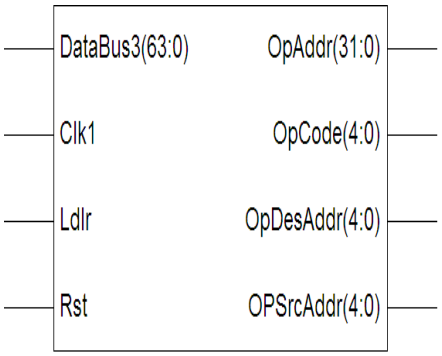


Fig.2: Block Diagram of Instruction Register

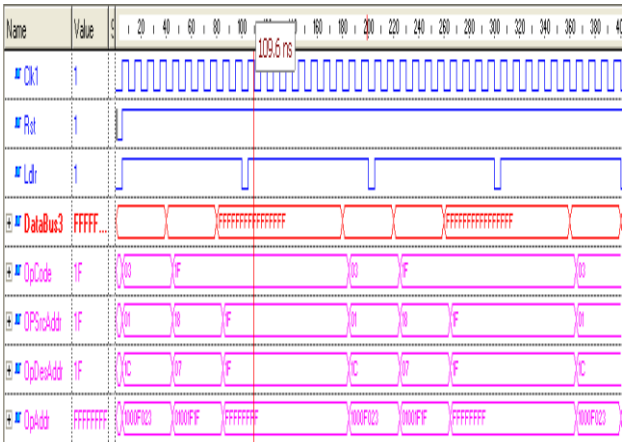


Fig.3: Simulation Results

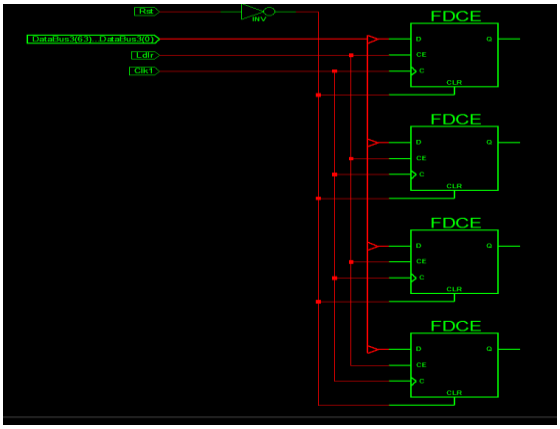


Fig.4: Technology schematic

Table 2. Implementation Results

Logic Utilization	Usage	Availability
Slices	**1**	**768**
Flip Flops	**47**	**1536**
LUTs	**1**	**1536**
IOBs	**93**	**124**

Arithmetic Logical Unit: The arithmetic logical unit has 16 operations each one of them was created and converted in to a symbol, and then a multiplexer was placed in order to obtain a 4-bit selector. Below depicts the Block diagram of ALU, Simulation results and Technology schematic and a table of Implementation results.

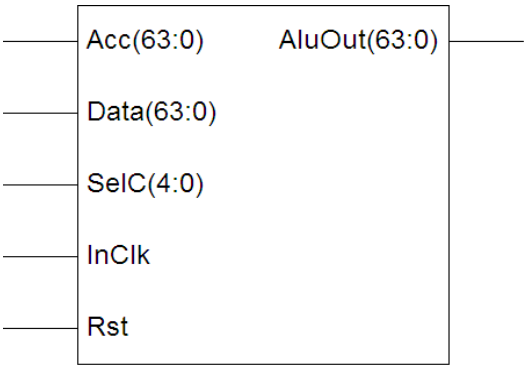


Fig.5. Block diagram of ALU

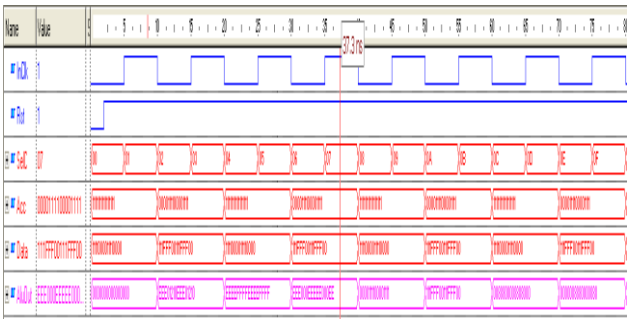


Fig.6: Simulation Results

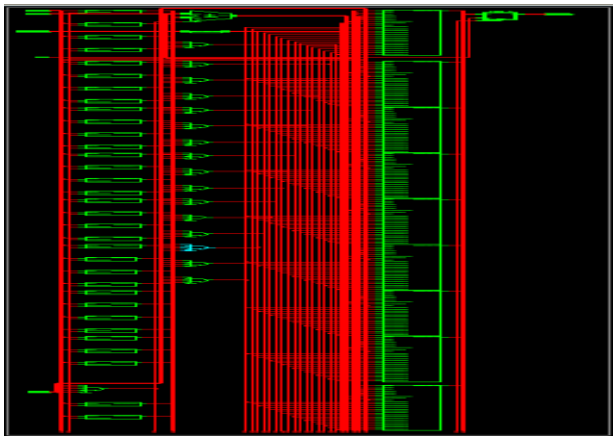


Fig.7. Technology Schematic

Table.3: Implementation Results

Logic Utilization	Usage	Availability
Slices	**360**	**768**
Flip Flops	**64**	**1536**
LUTs	**652**	**1536**
IOBs	**199**	**124**

General Purpose Registers: General purpose registers store and save operands and results during program execution. ALU and memory must be able to write/read those registers so a set of sixteen 32-bit registers were used along with multiplexers and control circuit which are the operands to ALU which perform the operation. Below shows the Block diagram of GPR, Simulation Results, Technology Schematic and a table of Implementation Result.

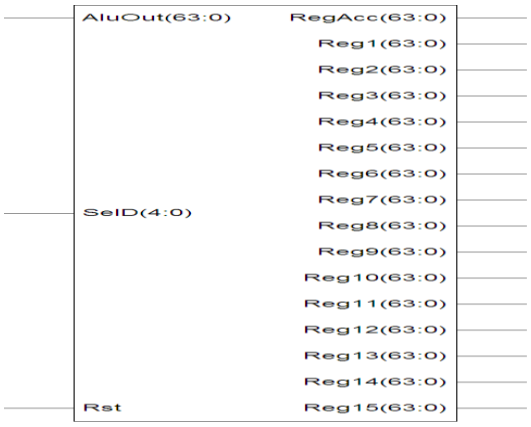


Fig.8: Block Diagram of GPR

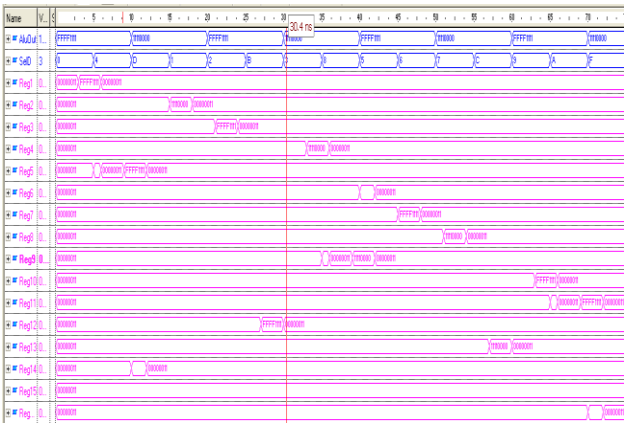


Fig.9: Simulation Results

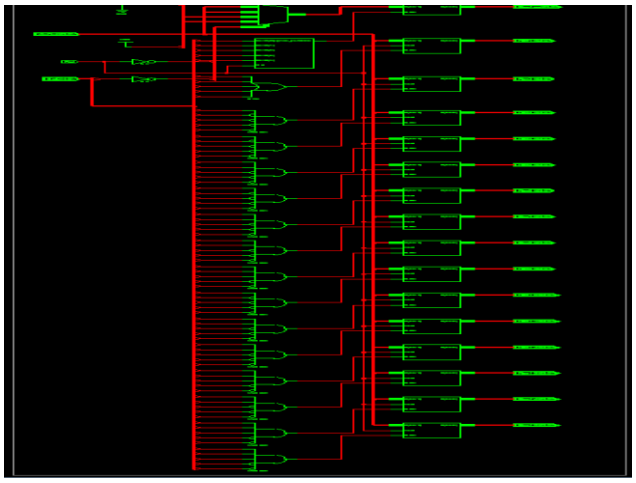


Fig.10: Technology Schematic

Table.4: Implementation Result

Logic Utilization	Usage	Availability
Slices	**48**	**768**
Flip Flops	**87**	**1536**
LUTs	**1024**	**1536**
IOBs	**8**	**124**

Control Unit: The control unit is based on using FSM 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 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 memory and 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. Below shows the Block diagram of Control unit, Simulation Results, Technology Schematic and a table of Implementation Results.

Fig.11: Block Diagram of Control unit

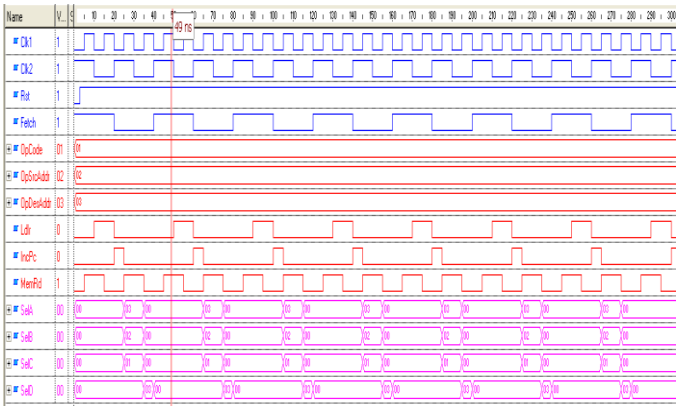


Fig.12: Simulation Results

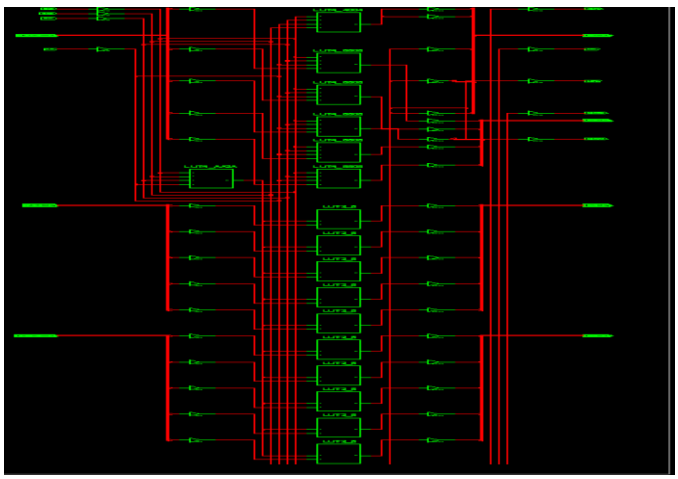


Fig.13: Technology Schematic

Table.5: Implementation Results

Logic Utilization	U sage	Availability
Slices	12	768
Flip Flops	44	1536
LUTs	20	1536
IOBs	44	124

V. CONCLUSION

The proposed research work, 32-bit cryptographic processor performs mathematical computations used in symmetric key algorithms (AES, DES, Blowfish, IDEA), has been designed using Verilog HDL. This 32-bit processor introduces the cryptographic instructions like Modular addition, Polynomial matrix multiplication, Reduction Modulo Multiplication, and Rotate word. The simulations are performed using standard Active HDL simulator and implementation is carried out using Xilinx ISE tool. 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 power consumption. In order to obtain a more sophisticated architecture is necessary to add some advanced techniques.

REFERENCES

[1]. Jun-Hong Chen, Ming-DerShieh, "A High-Performance Unified Field Reconfigurable Cryptographic Processor", IEEE TRANSACTIONS on very large scale integration (VLSI) systems, vol.18, issue 8, PP:1145-1158, 2010.
[2]. D. Mandalidis, P. kenterlis, J. Ellinas, "A Computer Architecture Educational System based on a 32-bit processor" International review on Computers and software, pp.114-119, 2008.
[3]. Antonio H. Zavala "RISC-Based Architecture for Computer Hardware Instruction" Edicion, Vol.1, issue 1, july 2012, 2011.
[4]. "Data Encryption Standard". FIPS publication, 1999 october 25.
[5]. "Advance Encryption Standard". FIPS publication 197, November 26, 2001.
[6]. M. Jaumain, et. al., " Educational simulation of the RISC processor," ICSE International Conference on Engineering Education, 2007.
[7]. J. Djordjevic, et. al., "An integrated Environment for Teaching Computer Architecture," IEEE Micro Vol. 20, Issue 3, pp. 66-74, 2000.
[8]. M. Morris Mano, Computer System Architecture, Prentice-hall, 1993.
[9]. M. Matsui, "On correlation between the order of S-Boxes and the strength of DES," precedings of Eurocrypt 94 (A. De Santis, ed.),no. 950 in lecture notes in Computer Science, pp.366375, Springerverlag, 1995.