# DESIGN A HIGH SPEED A TERNARY MULTIPLIER FOR CRYPTOGRAPHIC SYSTEM

[1]P.VIJAYALAKSHMI, [2]G.MALYADRI
[1]M.tech-Scholar, Dept of ECE, KKR&KSR institute of technology and sciences, Guntur, A.P, India
[2]Associate Professor, Dept of ECE, KKR&KSR institute of technology and sciences, Guntur, A.P, India

**ABSTRACT: Cryptography is the study of techniques for secure communication in the presence of third parties called adversaries. Generally, cryptography is about constructing and analyzing protocols that prevent third parties or the public from reading private messages. Modern cryptographic techniques have multiple applications, like access control, for electronic money transfers, for copyright protection as well as digitally sign documents. Since the usages are highly vital, users need to check the efficiency of the cryptographic techniques. Major part of cryptography is to design and implement Encryption, a means to convert meaningful messages into total meaningless message. If the message is not encrypted, anyone would be able to read and monitor anyone's messages. Various algorithms have been proposed for encryption and description in the past. In this paper, AES-128 modification is presented to increase the security. In proposed system, ternary multiplier is used for fast of operation. Simulation results show the accuracy of the algorithm.**

**KEYWORDS: Encryption, Decryption, Round Key, Ternary Adder.**

## I. INTRODUCTION

Digital computer systems that are now used are binary digital systems. But, there is a new logic which is making its way to a new future. The logic which unlike binary uses 3 symbols. A number system built with a radix 3 is called as ternary number system. Though binary circuits are easy to implement in CMOS technology, they have got their own limitations. One of the limitations is, the larger the operands, the more amounts of binary circuits needs to be cascaded. For an example consider the following case, design of a parallel adder for unsigned number.

If it is assumed that the maximum value of operand is always less than 256, it may be able to use an 8-bit adder, which is built by cascading eight 1-bit adders. And if it is assumed that the number can take value 64K, and then its needed to built a 16-bit adder, by cascading single bit adders.

In contrast to the above approach, we can greatly optimize my chip area, if we use ternary adder instead to binary. This is because of the huge code capacity of ternary logic. In ternary logic an 8-bit code can represent a number as large as 6K.(i.e. 3^8). Therefore, the above discussion suggests that ternary indeed can be a very good alternative to the binary. The only problem being, the circuit in ternary can't be that easily designed as in binary.

This fact again provides a good opportunity for developing new custom circuits. One such circuit is the one proposed in this paper. In fact, we can develop any combinational ternary circuit, if we could follow the ternary k-map suggested, but to do that we need to have a set of basic gates for ternary. Even if we are able to implement the circuit, it is going to be very huge, consisting of hundreds of transistor if not thousands. To get the same functionality with reduced number of transistor we need to develop each circuit customarily, a lot of gates had been proposed for ternary gates, we try to propose other combinational circuits.

The chip area increases when the number of functions increases and when the number of inputs increases in binary logic. To overcome these problems multi valued logic systems are used where the radix is greater than 2 are the main trust for research. Ternary logic is one of the multi valued logic system with base 3. For 'n' inputs the number of functions realized will be larger in ternary logic than in binary logic. So by using ternary logic area can be reduced for higher bit order.

## II. EXISTED SYSTEM

Ternary coded decimal (TCD) is required in electronic systems e.g. where a decimal value needs to be displayed. It is better to process the data in TCD format if input and outputs are decimal, than to convert it from decimal to ternary and back to decimal. By utilizing TCD codes, the manipulation of decimal data can be greatly simplified by treating each digit as a separate single sub-circuit. This matches much more closely the physical reality of decimal input and output hardware. If the numeric quantity were stored and manipulated as pure ternary, interfacing to such a display would require complex circuitry. Therefore, in cases where the calculations are relatively simple, working throughout with TCD can lead to a simpler overall system than converting to ternary.

### A. TCD codes of Decimal numbers:

TCD is another method to represent decimal numbers. Each decimal number is defined by a ternary code of 3 bits.

**TABLE I. TABLE OF TCD CODES FOR DIFFERENT DECIMAL NUMBERS**

| Decimal | Bit2 | Bit1 | Bit0 |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 |
| 2 | 0 | 0 | 2 |
| 3 | 0 | 1 | 0 |
| 4 | 0 | 1 | 1 |
| 5 | 0 | 1 | 2 |
| 6 | 0 | 2 | 0 |
| 7 | 0 | 2 | 1 |
| 8 | 0 | 2 | 2 |
| 9 | 1 | 0 | 0 |

i.e. Only 0-9 has the correct TCD code.
When two TCD numbers are added Decimal numbers greater than nine may be obtained and hence the resulting code will not be a valid TCD. In such case we need to correct the result.

**B. Method for correcting the sum after addition:** After an analysis of the results it was found that the TCD sum needs the following correction process. If result is greater than 9, then correct by adding '122' to the obtained sum, to get a valid TCD output. Then carry produced will act as a higher digit.

If the sum is less than or equal to 9 then there is no need to add or modify it. If the sum is greater than 9 then we have to add ternary value '122'. Let us consider an example, addition of 6 (ternary form is 020) and 5(ternary form is 012) is 11( ternary form is 0102) which is greater than 9. So, we have to add ternary value '122' to the obtained result. Then the result is 1001. In this result, 001 represent first bit (1 in decimal no). Remaining carry '1' represent second bit (1 in decimal no).

**C. TCD Architecture:** The proposed architecture of the TCD adder is shown in the figure below.

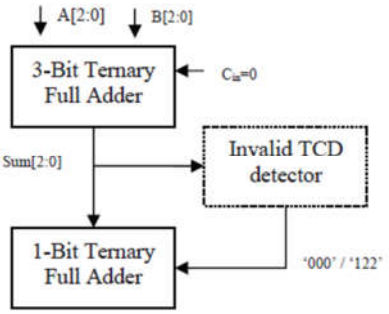


**Fig. 1. Ternary TCD Adder Block diagram**

This is the basic block diagram of the ternary TCD adder and it shows the different block involved in TCD adder. Each block is designed and simulated in cadence. Later all the blocks are interfaced to get the TCD adder. A and B are two ternary input and Cin is the carry input. The 'Sum' output of the first adder may or may not be a valid TCD code. An invalid TCD code is detected by an 'Invalid TCD detector circuit'. If the code is invalid it gets modified by the next adder preset in the data path.

## III. TERNARY HALF-ADDER

**A. Ternary half Adder:** The truth table of a ternary half adder circuit is given by the following table II. A and B represent two input which can take three different values 0, 1, 2. The output will also consist of three levels. The expected result from the resulting nine combinations is shown in table II

**TABLE II. TRUTH TABLE OF TERNARY HALF ADDER**

| A | B | Sum | Carry |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 0 | 2 | 2 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 2 | 0 |
| 1 | 2 | 0 | 1 |
| 2 | 0 | 2 | 0 |
| 2 | 1 | 0 | 1 |
| 2 | 2 | 1 | 1 |

This functionality can be implemented in a variety of ways. This work uses a less transistor count half adder. An optimized ternary half adder circuit is used in this work.

## IV. DESIGNING OF TERNARY FULL-ADDER

**A. Ternary Full Adder:** The expected truth table of a ternary full adder circuit is given by the following table iii. A, B, C in represent two input which can take three different values 0, 1, 2. The output will also consist of three levels. The expected result from the resulting 27 combinations is shown in table.iii.

**TABLE III. TRUTH TABLE OF TERNARY FULL ADDER**

| A | B | Cin | Sum | Carry |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 0 | 2 | 2 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 2 | 0 |
| 0 | 1 | 2 | 0 | 1 |
| 0 | 2 | 0 | 2 | 0 |
| 0 | 2 | 1 | 0 | 1 |
| 0 | 2 | 2 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 0 | 2 | 2 | 0 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 2 | 0 |
| 1 | 1 | 2 | 0 | 1 |
| 1 | 2 | 0 | 2 | 0 |
| 1 | 2 | 1 | 0 | 1 |
| 1 | 2 | 2 | 1 | 1 |
| 2 | 0 | 0 | 0 | 0 |
| 2 | 0 | 1 | 1 | 0 |
| 2 | 0 | 2 | 2 | 0 |
| 2 | 1 | 0 | 1 | 0 |
| 2 | 1 | 1 | 2 | 0 |
| 2 | 1 | 2 | 0 | 1 |
| 2 | 2 | 0 | 2 | 0 |
| 2 | 2 | 1 | 0 | 1 |
| 2 | 2 | 2 | 1 | 1 |

This functionality can be implemented in a variety of ways. Firstly, we can use the ternary K-map suggested. But by using it we find that the number of gates and hence the number of transistors drastically

increases. The other alternative is to design a custom circuit to get the above functionality.

**B. Ternary Full Adder using Ternary Half adder:** As TCD Adder circuit is built using Ternary full adders, therefore full adder circuit needs to be developed first. An optimised way to implement full adder is, with the help of half adder than that of implementing full adder itself by using k map methods. The implementation is as shown in the fig.3.

Minimum of 3 half adders are utilised to design a complete ternary full adder. After adding A and B its sum is given to the next half adder as an input along with C. Its sum is the ultimate, full adder sum. The carry generated by Half adder( using A,B) and carry generated from the second adder for which C is given as one of the inputs are added with the help of half adder. The sum obtained by those two carries is the final carry of the full adder.

## V. PROPOSED SYSTEM

**Encryption:** Initial step in any cryptographic algorithm is receiving plaintext messages and converting it to desired format as is necessitated by the algorithm. On receiving the data, plaintext formats it into a 4x4 matrix and operates through various processes. The first step, the user is asked for the message to be encrypted and the cipher key to be used in the algorithm. In the next step, the input plaintext has to be blocked in a specified format. AES is a block cipher algorithm with a block size of 128 bits, and whose various round of operations works with a format using 4x4 state matrix. Therefore in the very opening stage of AES, plaintext is converted to a 128 bit 4x4 state matrix.

The AES algorithm supports 128, 192, or 256 bit keys, which generates 128 bit intermediate round keys for each round of operation. The initial round key is the first 128 bits of the key used in round 0. The next round key is a transformation of the first round key. Other round keys are similarly generated by transformations of previous round keys.

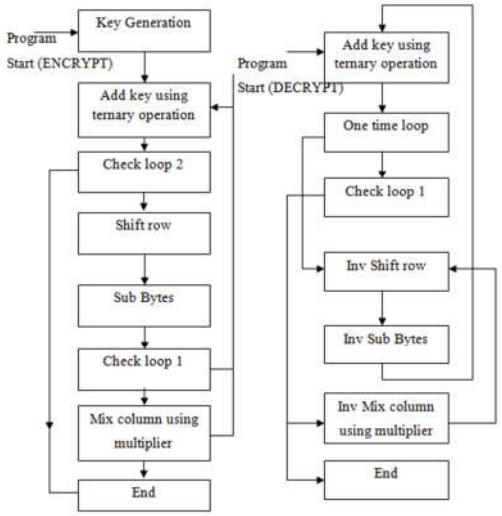


**Fig 2. Proposed System**

**Key Expansion Operation:**
Cyclic operation of AES initiates with this operation, which aids in generating 10 round keys in AES 128 bit variant cryptographic algorithm. A small algorithm in its own way, it calls for different sub-functions for creating confusion to a cipher key such that it will not be traced or attacked by an intruder. The various functions called in this operation strengthens the key and expands it such a way that every time a different version of the cipher key is available for each iterative round of AES encryption algorithm. If the same key is used for every iterative step, an attacker finds it easier to encrypt/decrypt

the whole message, if it somehow has access to the key. Functions call for rotation, substitution and addition operation for creating diffusion.

**Add Round Key Operation:** During pre-round transformation, the initial state matrix and expanded key is XORed in add round key transformation to produce the next state matrix.

**Sub Byte Operation:** One of the main ingredients in this step is the formation of an S-Box.
The next transformation is sub_byte operation, wherein each element of the previous state is replaced by a new element from a substitution table. The next state from this operation is shown.

**Row Shift Operation:** The next transformation is shift_row operation, wherein each element of the previous state is cyclically shifted to its left in the prerequisite order and form the next state.

**Mix Column Operation:** The next transformation is Mix_Column operation, wherein each row of the previous state is multiplied with a mix multiplication constant matrix to form the next state.

These operations continue iteratively for 10, 12 and 14 rounds with respect to 128, 192 and 256 bits of key length in various AES variants and a constant 128 bit block length.

**Decryption:** When an authorized user receives the encrypted message, with the help of a secretive symmetric key, original message can be extracted out from the cipher text. This cipher text is passed through AES decrypting transformation of 10 rounds before this gibberish message is decrypted to a meaningful message. The last round of 10 transformations is displayed with results.

Since AES is a symmetric cipher, it will be using same cipher key for encryption and decryption both. For decrypting, the fed cipher key is taken into process for different rounds of operation in inverse order. Since in this decryption, last round of operation is shown, which is almost similar in action to the first round of operation in encryption side (except for Inverse MixColumn operation).

**Inverse Shift Row:** The next transformation is Inv_shift_row operation, wherein each element of the previous state is cyclically shifted to its right in the prerequisite order and form the next state.

**Inv Sub Byte:** The next transformation is Inv sub byte operation, wherein each element of the previous state is replaced by a new element from an inverse substitution table, which is derived from the substitution table.

**Add Round Key:** Following the algorithm, last round of AES 128 bit variant, takes as input the 16 bit hexadecimal cipher key, passes it through Key Expansion transformation and gives a 4x4 matrix of the cipher key characters.

During last round transformation, the initial state matrix and expanded key is XORed in the add round key transformation to produce the next state matrix. Finally the original message is derived from the ciphered message using the ciphered key and the cipher text, with this final round of AES transformations.

This text is displayed to the authorized receiver who is having the correct cipher

key for decrypting the ciphered message, let alone an intruder who is trying to capture the key and eventually the message.

## VI. RESULTS

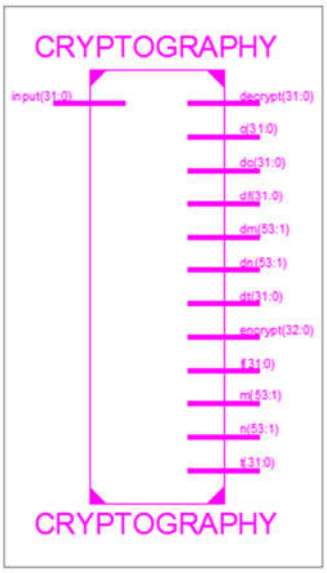


**Fig 3. RTL Schematic**

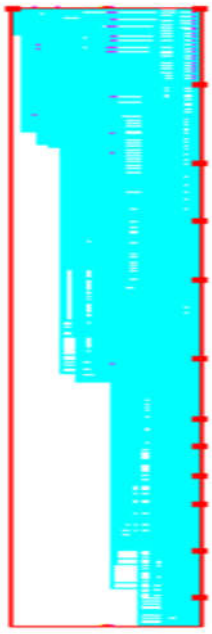

**Fig 4. Technology Schematic**

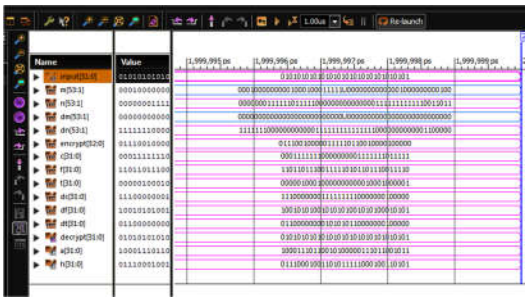

**Fig 5. Output**

## VII.CONCLUSION

In this paper, we propose an efficient VLSI architecture for advanced encryption standard design methodology in order to provide a high-speed and effective cryptographic operation. High-performance and fast implementation of proposed ternary multiplication is applied to cryptographic systems. Cryptography is the operation in wireless communication between transmissions and receiving of data, the secured data is communicated in an unsecured channel between transmitter and receiver with high security. The total proposal is done in XILINX 14.7 with Spartan 3E family.

## VIII. REFERENCES

[1]. Manjesh K N and R K Karunavathi ," Secured High throughput implementation of AES Algorithm ", International Journal of Advanced Research in Computer Science and Software Engineering, vol.5 pp. 1193- 1198, May 2013 .

[2]. Hoang Trang and Nguyen Van Loi, "An efficient FPGA implementation of the Advanced Encryption Standard (AES) algorithm", IEEE Transactions, vol. 20 pp. 978-1-4673-0309-5, 2012.

[3]. AJ Elbirt, W Yip, B Chetwynd and C Paar, "An FPGA Based Performance Evaluation of the AES Block Cipher

Candidate Algorithm Finalists", IEEE Transactions on VLSI, 2010.
[4]. G.P. Saggese, A. Mazzeo, N. Mazzocca and A.G.M. Strollo. An FPGA-Based Performance Analysis of the Unrolling, Tiling, and Pipelining of the AES Algorithm, *in FPL 2003*, LNCS 2778, pp. 292- 302, 2003.
[5]. KimmoJarvinen, MattiTommiska and JormaSkytta, "Comparative Survey of High Performance Cryptographic Algorithm Implementations on FPGAs", IEEE Proceedings - Information Security, vol. 152, no. I, Oct. 2005, pp. 3-12.
[6]. Tim Good and MohannedBenaissa "Very Small FPGA Application-Specific Instruction Processor for AES," IEEE transactions on circuits and systems, vol. 53, issue. 7, pp. 1477-1486, July 2006.
[7]. Xinmiao Zhang and Keshab K. Parhi "High-Speed VLSI Architectures for the AES Algorithm," IEEE transactions on very large scale integration (vlsi) systems, vol. 12, ISSUE. 09, pp. 957-967, September 2004.
[8]. Adam J. Elbirt, W. Yip, B. Chetwynd and C. Paar, "An FPGA-Based Performance Evaluation of the AES Block Cipher Candidate Algorithm Finalists," IEEE transactions on very large scale integration (vlsi) systems, vol. 9, issue. 4, pp. 545-557, August 2001.
[9]. P. S. Abhijith, M. Srivastava, A. Mishra, M. Goswami and B. R. Singh, "High Performance Hardware Implementation of AES using Minimal Resources," IEEE International Conference on Intelligent Systems and Signal Processing, Gujarat, pp. 338-343, March 2013.
[10]. Trang Hoang and Van Loi Nguyen, "An Efficient FPGA Implementation of the Advanced Encryption Standard Algorithm," IEEE International Conference on Computing and Communication Technologies, Research, Innovation and Vision for the Future, Ho Chi Minh City, pp. 1-4, February-March 2012.

**P. VIJAYA LAKSHMI** completed her B.TECH at KKR&KSR institute of technology and sciences, Guntur. She is pursuing her M.TECH in KKR&KSR institute of technology and sciences, Guntur. Her Specialization is VLSI.

**G.MALYADRI** working as Associte Professor in the Dept. of ECE, KKR&KSR institute of technology and sciences, Guntur. He has 16 years of experience. His area of interest is in Image Processing. He has membership in ISTE.