HIGH PERFORMANCE MONTGOMERY MULTIPLICATION USING DADDA TREE ADDITION

Thandri Adi Varalakshmi Devi[1], P Subhashini [2]
[1]PG Scholar, Dept of ECE, Kakinada Institute of Technology, Korangi, AP, India.
[2]Assistant Professor, Dept of ECE, Kakinada Institute of Technology, Korangi, AP, India

Abstract
In recent years, finite fields arithmetic over GF(2^m) has gained very high importance due to its application in elliptic curve cryptography (ECC) to realize robust cryptosystems in resource-constrained environments. A number of architectures have been proposed for efficient implementation of multiplication over GF(2^m). Several algorithms and architectures are suggested in the literature for polynomial basis multiplication for the fields generated by trinomials and pentanomials, primarily due to their computational simplicity. In this, the decomposing of the Dadda multiplication into two concurrent blocks and we have derived a lower-latency multiplier using the proposed modular reduction scheme using PCA. From the table it can be conclude that the proposed Dadda multiplier along with Spanning tree adder gives better results than the existed Montgomery multiplier. This design which is proposed has significantly less area as well power complexities beyond with the best of the existing designs. For solving this we have used ISE simulator in verifying functionality and the synthesis was performed by using Xilinx ISE 12.3i EDA tool along with Verilog HDL coding Language.

Keywords: Pentanomial, Montgomery Multiplier, Finite Field, Verilog.

I. INTRODUCTION

A Finite field, which is known as a Galois field, that has been originated from the French mathematician Pierre Galois. The elements which will take q different values will be known as GF (q). Addition and multiplication were the two formal properties present for finite field. The outcome of two elements by adding or multiplying, present in the the field will always has an element located in the field itself. Any of the one element located in the field will be zero, so that a + 0

= a, if we take any element 'a' in the field. One of the element of the field is unity, so a • 1 = a for any element a in the field. For every element a in the field, there is an additive inverse element -a, such that a + (- a) = 0. It also allows the operation of subtraction which is defined as inverse of the addition. Each of non-zero element b in the field will have a multiplicative inverse element b-1 such that b b-1= 1. It also allows division operation of which it is defined as inverse of the multiplication. The associative [a + (b + c) = (a + b) + c, a

- (b • c) = [(a • b) • c], commutative [a + b = b + a, a • b = b • a], and distributive [a • (b + c) = a • b

+ a • c] laws apply. These properties cannot be satisfied for all possible field sizes. However these can be satisfied if the field size is of any prime number or any integer which should be power of a prime.The finite fields are classified as follows:

- The number of elements in a finite field is of the form p^n, where p is a prime number called the characteristic of the field, and n is a positive integer.
- For every prime number p and positive integer n, there exists a finite field with p^n elements.

By this it justified that finite fields specifies only the order of the field. One notation for a

finite field is $\mathbb{F}_{p^n}$ or Fpn. Another notation is GF(p^n), where the letters GF stand for "Galois field". A field of prime power with p = 2 is also known as binary field. First we consider fields where the size is prime, i.e., with n = 1. Such field is named a prime field, and is canonically isomorphic to the ring Z/pZ, the set of integers modulo p. Even though all fields which are of size p are isomorphic to Z/pZ, for n ≥ 2 the ring of integers modulo pn, Z/pnZ, is not a field. The element p is nonzero also it has no multiplicative inverse modulo pn. By comparison with the ring Z/4Z of size 4, the underlying additive group of the field (Z/2Z)[T]/(T2 + T + 1) of size 4 is not cyclic, even it is isomorphic to the Klein four-group. No fields exist for which the size is not a prime power. For example, there is no field with 6 elements. Given a prime power q = pn, explicitly we would construct a finite field with q elements as follows. Select a monic irreducible polynomial f(T) of degree n in Fp[T]. Then Fp[T]/(f(T)) is a field of size q Here, Fp[T] denotes the ring of all polynomials in T with coefficients in Fp,(f(T)) denotes the ideal generated by f(T), and the quotient is meant in the sense of quotient rings the set of polynomials in T with coefficients in Fp mod (f(T)).

II. MONTGOMERY MULTIPLICATION

In this type of multiplication method, the given values will be computed as 'ab mod m' for every positive integers a, b, and m. The execution time will be reduced when there will be of huge number of multiplications has to be performed along the same modulus m, and by small number of multipliers, using this multiplier. As a result, it is helpful in computing a^n mod m for a huge value of n. The count of multiplications of a modulo m in the above type can be reduced to a small number substantially from that of n by successively squaring combined with multiplying as per the pattern of the binary expression bits for n. But still this can be enough as a large enough number to be worthwhile speeding up if there is a possibility. The method, will changes the reduction modulo m to a reduction modulo r essentially. In general r is chosen as an integral power of 2, and then the reduction modulo r is simply a masking operation; which is, retaining the lg(r) low-order bits of an intermediate result, and discarding all higher order bits. If r is a power of 2, then m is odd, to satisfy the gcd requirement. (Any of the odd value from 3 to r − 1 would be acceptable.). Working on n-digit numbers to base d, a Montgomery step calculates $axb \div d^m \pmod r$

The base d is typically 2 for microelectronic applications, 2^8 for 8-bit firmware,[4] or 2^{32} or 2^{64} for software applications. Due to exposition, we shall illustrate with d = 10 and n = 4. To change this to a modular operation with a modulus r, add, immediately before every shift, whatever the multiple of r is to be needed for making the value in the accumulator a multiple of 10. The result will be, which means the final value in the accumulator would be an integer (since only multiples of 10 have been divided with 10) along with equivalent (modulo r) to 472× a ÷ 10000. Finding the appropriate multiple of r is the simplest operation for single-digit arithmetic. The theory as well practice for Montgomery multiplication has explained in this note. This will reduces execution-time a computer whenever there will be a huge number of multiplications are to be completed with the same modulus m, and by using less number of multipliers. In particular, it is useful for computing $a^n \pmod m$ for a huge value of n. The reductions of modulo m are the only difficulty in which, essentially division operations that are costly in execution time. If one defers the modulus operation till the end, so that the products may grow to very large numbers, which will slows down the multiplications and also the final modulus operation. For using Montgomery multiplication, the multipliers a and b must have less than the modulus m. Another integer r has been introduced by us. This method accordingly, changes the reduction modulo m to a reduction modulo r. In general r is chosen to be an inte-gral power of 2, thus the reduction modulo r is simply a masking operation, which means, retaining the lg(r) low-order bits of an intermediate result, and discarding higher order bits. If r is a power of 2, we must have m odd, for satisfying the gcd requirement.

III. GALOIS FIELD IN CRYPTOGRAPHY

In this paper we introduce the basics of Galois Field and also its implementation as of storing data. This paper shows and helps in visualizing as that of storing data in Galois Fields allows

manageable and effective data manipulation, where it focuses mainly on application in computer cryptography. Details about the algorithm used in Advanced Encryption Standard (AES), which is an example for computer cryptography that utilizes Galois Field, will also be included. Galois Field, named after Evariste Galois, also known as finite field, refers to a field in which there exist finitely many elements. It is particularly useful in translating computer data as they are represented in binary forms.

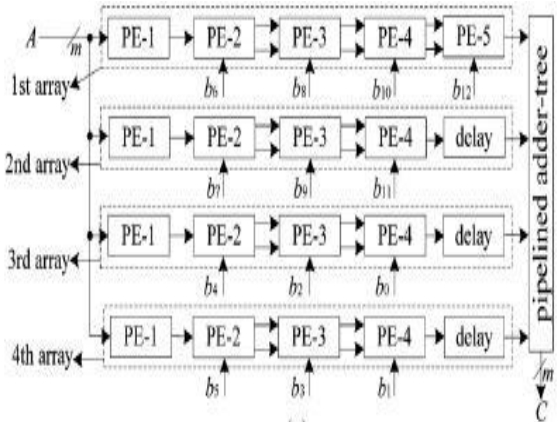

Fig1. Low Latency Montgomery Multiplier Design.

Binary System: In the binary numeral system or base-2 number system, we represent each value with 0 and 1. To convert a decimal numeral system or base-10 number system into binary system, we need to represent a decimal in terms of sums of an2^n.

$$x = \sum_{n \in \mathbb{N}} a_n 2^n$$

For leading and omitting zeros the coefficients an is then written in descending. Thus produced final value will be the decimal equivalent. Ultimately, binary system offers an alternative way of representing the elements of a Galois Field.

Example:

$$19 = \ldots + 1 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0$$

So the binary representation of 19 will be as 10011 and where as the elements of gf(2^3) in binary are gf (2^3) = (001, 010, 011, 100, 101, 110, 111).

Dadda Multiplier: The Dadda multiplier is a type of hardware multiplier which is a type of design, invented by computer scientist known as Luigi Dadda in 1965. Which is similar to that of Wallace multiplier, but not exactly and it is of little faster (for all operand sizes) and requires less number of gates (for all but the smallest operand sizes). However, Wallace multipliers are not same as like Dadda multiplier, which reduces as much as possible accordingly on each and every layer. Because of this reason, Dadda multipliers is a little expensive reduction phase, However the numbers may be of few bits longer, So as requiring slightly bigger adders. For getting this, the structure of the second step is modified by slightly with complex rules than in the Wallace tree. As of the Wallace tree, a new layer would be added as if the weight is carried by three or else with more wires. The rules in the reduction for the Dadda tree, however, are as shown below:

- Take any of the three available wires which are of same weights and give those as inputs into a full adder. The final result wil output wire of the same weight and an output wire which are of higher weight for each of the three input wires.
- If there are two wires were left and are of the same weight, and also as the current number of output wires as of the weight is equal to 2 (modulo 3), give as input into a half adder. Or else, bypass them through to the next layer.
- If there were only one wire left, then connect it to the next layer.

In this step addition will be only be done as many times as required as per the need, so that the output weights count will stays close to the multiple of 3, which is the ideal count of weights when-ever we use full adders as of 3:2 compressors. If two wires were present which are of same weight, left it them as it is, and the number of output wires at that time with that weight would be equal to 1 or 2 (modulo 3), input them into a half adder. Or else, bypass them to the next layer.

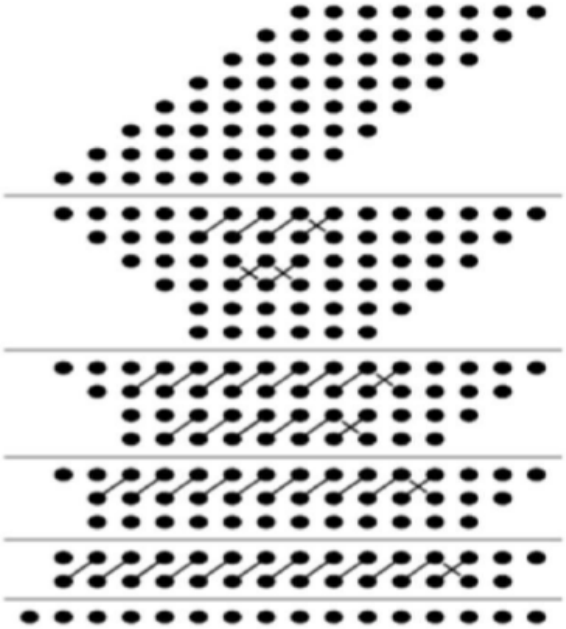

Fig2. Dot Diagram for 8 by 8 Dadda Multiplier.

Spanning Tree Adder: We can state that an undirected graphs which are having a set of V of vertices (also can represented with nodes) and a set E of edges, each of which is connecting two separate vertices. More by mathematically calculations, we would speak that the edge relation between vertices is of same manner for undirected graphs. Here we discuss about undirected graphs, as if directed graphs will also play an important role in most of the applications. Below is the simple example for a connected, undirected graph along with 5 vertices (A;B;C;D;E) and 6 edges (AB, BC, CD, AE, BE, CE)

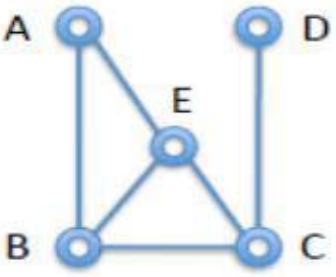

Fig3. Examples of a connected, undirected graph with 5 vertices.

Here we were interested particularly towards the problem of computing a spanning tree for a connected graph. What is a tree here? They are nothing but a bit different than the binary search trees which are considered previously. One of the definition which is simple was that a tree is a connected graph consists with no cycles, where a cycle let's you go from a node to itself without repeating any of its edge. A connected graph for a spanning tree, represented with G is a tree containing all the vertices of G. The two examples of spanning trees were represented below, which reflects our original example graph.

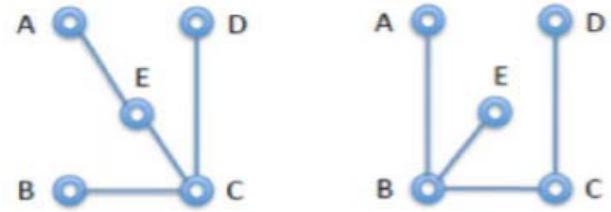

Fig4. Examples of spanning trees for our original.

When a asymptotic complexity is considered, it is necessary to categorize graphs with a dense or sparse. A lot of edges will be present in dense graphs when compared to the number of vertices. Representing the n = |V| for most of the vertices, as we know there can be at most n * (n-1)/2: each and every node is connected to any of the other node (n * (n-1)), but in an undirected way (n*(n-1)/2). If we represent e as the number of edges, we would have e = $O(n_2)$. By comparison, a tree is sparse because e = n-1 = O(n).

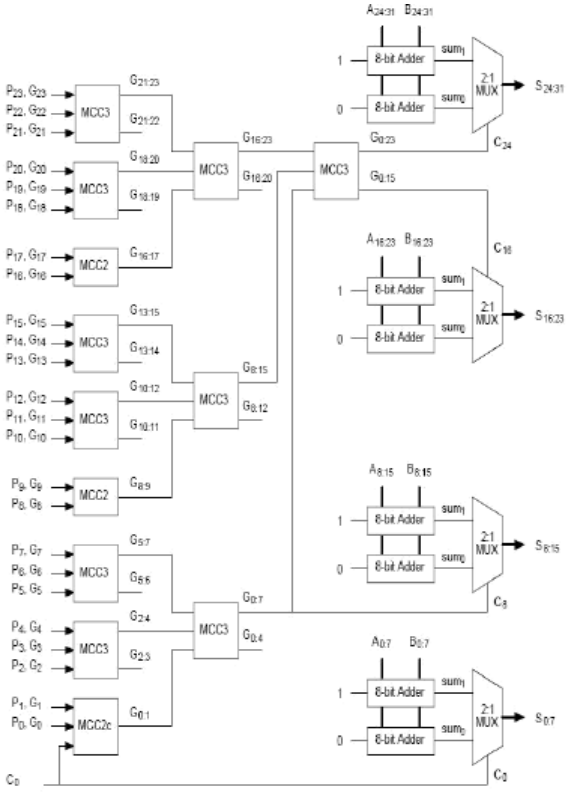

Fig5. Spanning Tree Adder Architecture.

IV. RESULTS

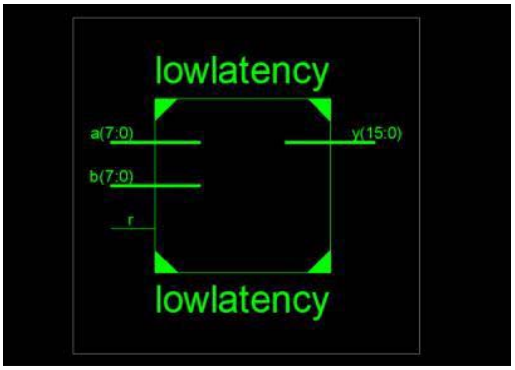

Fig6. RTL.

Fig7. Technological

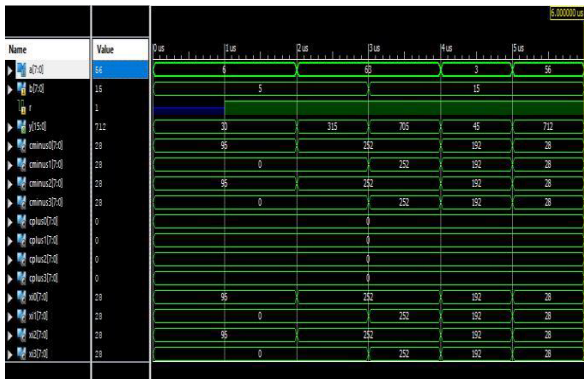

Fig9. Simulation Waveform.

A. Table1. Comparison

	Number of 4 input LUTs	Power Consumption (mW)	Delay (ns)
Existing System	**153**	**1.248**	**18.831**
Proposed System	**121**	**0.987**	**30.417**

V.CONCLUSION

In this project, we have presented a novel PCA technique and modular reduction scheme for Montgomery multiplication over GF(2m) based on irreducible pentanomials. To illustrate the efficiency of the proposed approach, it has designed the multiplier for the irreducible pentanomial for simplicity of proceeding. In this, the decomposing of the Dadda multiplication into two concurrent blocks and we have derived a lower-latency multiplier using the proposed modular reduction scheme using PCA. From the table it can be conclude that the proposed Dadda multiplier along with Spanning tree adder gives better results than the existed Montgomery multiplier. The multiplier design proposed in this paper produced 121 LUT'S whereas the conventional multiplier produced 10490 LUT's. This indicates that the proposed multiplier design in area efficient. Power consumed by the Montgomery multiplier is 9.196 mw which is less than conventional multiplier design having 797.24 mw of power consumption. Because in the proposed Montgomery design the processing elements will reduce the maximum logic when compare with the other multipliers which are the main advantage of this design.

VI. REFERENCES

[1] R.L. Rivest, A. Sharmir, and L. Adleman, "A Method of Obtaining Digital Signature and Public-Key Cryptosystems," Comm. ACM., vol. 21, no. 2, pp. 120-126, 1982

[2] W. Diffie and M.E. Hellman, "New Directions in Cryptography," IEEE Trans. Information Theory, vol. 22, pp. 644-654, 1976 '

[3] P.L. Montgomery, "Modular Multiplication without Trial Division," Math. Computing, vol. 44, no. 170, pp. 519-521, Apr. 1985.

[4] J. Menezes, P.C. van Oorschot, and S.A. Vanstone, Handbook of Applied Cryptography, CRC Press, 1997.

[5] C. Walter, "Systolic Modular Multiplication," IEEE Trans. Computers, vol. 42, no. 3, pp. 376-378, Mar. 1993 .

[6] P. Kornerup, "A Systolic, Linear-Array Multiplier for a Class of Right-Shift Algorithms," IEEE Trans. Computers, vol. 43, no. 8, pp. 892-898, Aug. 1994.

[7] W.C. Tsai, C.B. Shung, and S.J. Wang, "Two Systolic Architecture for Modular Multiplication,", IEEE Trans. VLSI, vol. 8, no. 1, pp. 103-107, Feb. 2000.

[8] A.F. Tenca and C.K. Koc, "A Scalable Architecture for Modular Multiplication Based on Montgomery's Algorithm," IEEE Trans. Computers, vol. 52, no.9, pp. 1215-1221, Sep. 2003.

[9] A.F. Tenca and C.K. Koc, "High-Radix Design of a Scalable Modular Multiplier," Proc. Cryptographic Hardware and Embedded Systems (CHES 2001), pp. 189-205, May 2001.

[10] J.J. Leu and A.Y. Wu, "Design Methodology for Booth-Encoded Montgomery Module Design for RSA Cryptp-system," Proc. IEEE Int. Symp. Circuits and Systems (ISCAS-2000), pp. V.357-360, May 2000.

[11] J.H.Hong and C.W.Wu,"Radix-4 Modular Multiplication and Exponentiation Algorithm for the RSA Public-key Cryptosystem," ASP-DAC, pp. 565-570, 2000 .

[12] S.F. Hsiao, M.R. Jiang and J.S. Yeh, "Design of high speed low-power 3-2 counter and 4-2 compressor for fast multipliers," Electronics Letters, vol. 34, no. 4, pp. 341-343, Feb. 1998 .

[13] H. Orup, "Simplifying Quotient Determination in High - Radix Modular Multiplication," Proc. 12th Symp. Computer Arithmetic, pp. 193-199, July 1995.