ISSN: 2278 – 1323
International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)
Volume 1, Issue 9, November 2012

Design and Implementation of Arithmetic Unit for GF(2^m)

O.B.B.Madhuri, E.Rambabu, Malijeddi Murali

Abstract -— In Abstract algebra, a Finite field or Galois field (so named in honor of Évariste Galois) is a field that contains only finitely many elements. Finite fields are important in number theory, Algebraic geometry, Galois Theory, Cryptography, and Coding theory.Arithmetic in a finite field is different from standard integer arithmetic. There are a limited number of elements in the finite field; all operations performed in the finite field result in an element within that field.An arithmetic unit (AU) that performs all basic arithmetic operations in the finite field GF(2^m) will be implemented, where m is an arbitrary integer. The finite field AU consists of an arithmetic processor, an arithmetic logic unit, and a control unit. The proposed AU has low circuit complexity and is programmable, so that any error-correcting decoder that operates in GF (2^m) can be easily implemented with this AU.

***Index Terms*—Arithmeticunit(AU),Error correcting codes,finite field**

I .INTRODUCTION

The Finite Fields play an important role in the structure of Error-correction codes .The codes with symbols from the binary field GF(2) or its extension GF(2^m) are most widely used because information in computer and digital communication systems is universally coded in binary form for practical reasons. **T**he Forward error-correction codes can be employed to improve the system performance and thus have been widely used in Computer and Digital communication systems, such as the Compact Disc (CD), Third-Generation (3G) mobile system, Digital Audio Broadcasting (DAB), Digital Video Broadcasting (DVB),Wireless Metropolitan Area Network (WMAN), etc. . To design an error-correction decoder with low circuit complexity, it is necessary to have a multifunction arithmetic circuit. Therefore, there is a trend, when designing the multifunction arithmetic circuit, to reduce its complexity, shorten its calculating delay and increase its operation speed. In a finite field GF(2^m), an arithmetic circuit with high-speed, low complexity and versatile features is required. Therefore, an arithmetic unit (AU) which can perform all basicc arithmetic operations in the finite field GF(2^m) is proposed in this paper. The present AU is structured with low circuit complexity, so that an error-correction decoder applying this AU can be greatly simplified.

II. BASIC THEORY OF FINITE FIELDS

Arithmetic in a finite field is different from standard integer arithmetic. There are a limited number of elements in the finite field; all operations performed in the finite field result in an element within that field.

O.B.B.Madhuri *,M.Tech VLSI 2'nd Year Student, (Electronics and Communications Engineering), Andhra University/ Sanketika Vidya Parishad Engineering College, visakhapatnam,india,+918143866065.*

E.Rambabu *, Assistant Professor, Electronics and Communications Engineering, Andhra University/Sanketika Vidya Parishad Engineering College, Visakhapatnam, India,+9173827673*

Malijeddi Murali*, Professor & HOD, Electronics and Communications Engineering, Andhra University/Sanketika Vidya Parishad Engineering College, Visakhapatnam, India, +919440149970.,* While each finite field is itself not infinite, there are infinitely many different finite fields; their number of elements (which is also called cardinality) is necessarily of the form p^m where p is a prime number and m is a positive integer, and two finite fields of the same size are isomorphic. The prime p is called the characteristic of the field, and the positive integer m is called the dimension of the field over its prime field.Addition and subtraction are performed by adding or subtracting two of these polynomials together, and reducing the result modulo the characteristic. In a finite field with characteristic 2, addition modulo 2, subtraction modulo 2, and XOR are identical. Thus,

Polynomial: $(x^6 + x^4 + x + 1) + (x^7 + x^6 + x^3 + x) = x^7 + x^4 + x^3 + 1$

Binary: {01010011} + {11001010} = {10011001}

Hexadecimal: {53} + {CA} = {99}

Notice that under regular addition of polynomials, the sum would contain a term $2x^6$, but that this term becomes $0x^6$ and is dropped when the answer is reduced modulo 2.

p_1	p_2	$p_1 + p_2$ (normal algebra)	$p_1 + p_2$ in GF(2^n)
$x^5 + x + 1$	$x^5 + x^2$	$2x^5 + x^2 + x + 1$	$x^2 + x + 1$
$x^4 + x^2$	$x^6 + x^2$	$x^6 + x^4 + 2x^2$	$x^6 + x^4$
$x + 1$	$x^2 + 1$	$x^2 + x + 2$	$x^2 + x$
$x^3 + x$	$x^2 + 1$	$x^3 + x^2 + x + 1$	$x^3 + x^2 + x + 1$
$x^2 + x$	$x^2 + x$	$2x^2 + 2x$	0

Table 2.3.1comparision between normal algebraic sum and the characteristic 2 finite field sum

Arithmetic Unit for Finite Field

Addition, multiplication, division, exponentiation and inverse multiplication are the most basic arithmetic operations in a finite field; and several kinds of circuits, e.g., multiplier, divider, inverter, exponentiator, power-sum circuit , etc., have been proposed to perform these arithmetic operations. However, these arithmetic circuits are, respectively, designed for a specific arithmetic operation, which is never enough, for example, for a forward error-correction decoder. In a finite field, a GF arithmetic circuit with high-speed, low complexity and versatile features is required. Therefore, an arithmetic unit (AU) which can perform all basic arithmetic operations in the finite field is proposed in this paper. The present AU is structured with low circuit complexity, so that an error-correction decoder applying this AU can be greatly simplified.

Let f(x) be an irreducible polynomial over GF(2) of degree m, i.e.,

$$f(x) = x^m + f_{m-1}x^{m-1} + f_{m-2}x^{m-2} + \ldots\ldots + f_1 x + 1 \quad (1)$$

where $f_i \in GF(2) = \{0,1\}$ and i=1,2,….,m-1. The finite field GF(2^m) is an extension field of the prime field GF(2) and consists of 2^m elements. Let the element α is a root of the irreducible polynomial $f(x)$. Then the modulo polynomial $x^m = f_{m-1}\alpha^{m-1} + f_{m-2}\alpha^{m-2} + \ldots\ldots + f_1\alpha + 1$ can be obtained when . By this, all nonzero elements of GF(2^m) can be represented in terms of the polynomial of degree with respect to the basis $\{1, \alpha, \alpha^2 \ldots \ldots \ldots \alpha^{m-1}\}$. The proposed AU in the finite field GF(2^m)

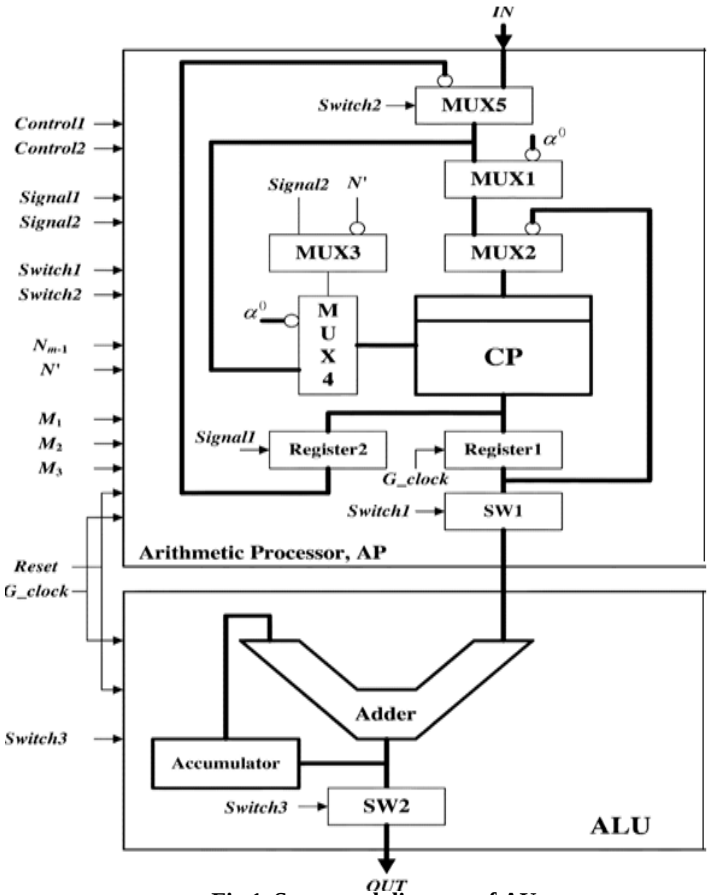


Fig.1. Structural diagram of AU.

includes an arithmetic processor (AP), an arithmetic logic unit (ALU) and control circuits. Therein the AP is structured on a calculating processor(CP) that can perform the $A \times B$ and $A \times B^2$ operations in the finite field GF(2^m) , where A and B are elements in the finite ield GF(2^m) . Based on this CP, multiplication, division, exponentiation and inverse multiplication can be performed . The major job of the ALU is provided to perform addition in the finite field GF(2^m). Adding the control circuits, all arithmetic operations in the finite field GF(2^m) can be completed using this AU. The architecture of the AU is shown in Fig.4.2.1

A. Calculating Processor (CP)

Based on the cellular-array power-sum circuit and the cellular-array multiplier, a CP is constructed. The CP is provided to perform $A \times B$ and $A \times B^2$ in the finite field , which includes an array of identity cells, as shown in Fig.4.3.1 Each identity cell includes three two-input AND gates, one two-input XOR gate, one three-input XOR gate, and a multiplexer, as shown in Fig.4.3.2

In this CP, what arithmetic operation which this CP wants to perform is decided by a control signal . Assume two input elements, A and B, are, respectively,

expressed as

$$A = a_{m-1}\alpha^{m-1} + a_{m-2}\alpha^{m-2} + \ldots\ldots + a_0 = [a_{m-1}, a_{m-2}, \ldots\ldots\ldots a_0] \quad (2)$$

$$B = b_{m-2}\alpha^{m-1} + b_{m-2}\alpha^{m-2} + \ldots\ldots + b_0 = [b_{m-1}, b_{m-2}, \ldots\ldots\ldots b_0] \quad (3)$$

Then, according to the irreducible polynomial and its modulo polynomial,

we have, $\propto^{m+1} = \alpha^m . \alpha = (f_{m-1}\alpha^{m-1} + f_{m-2}\alpha^{m-2} + \ldots\ldots + f_0).\alpha$

Multiplying $\alpha^m with\ \alpha$,and again subsuiting α^m in multiplication result,finally we get

$$= f_{m-1}^1\alpha^{m-1} + f_{m-2}^1\alpha^{m-2} + \ldots\ldots + f_0^1 \quad (4)$$

Where

$$f_i^1 = \begin{cases} f_{m-1}.f_0 & i = 0 \\ f_{m-1}.f_i + f_{i-1} & i = 1,2,\ldots\ldots m-1. \end{cases} \quad (5)$$

When the control signal ,control=0, $f_i^1 = 0, 0 \le i \le m-1$. and $A \times B^2$ operation is performed. This CP performs the $A \times B^2$ operation when control=1. At this time, $f_i^1 = f_{m-1}.f_i + f_{i-1}$ for $1 \le i \le m-1$ and . The outcome of this $f_0^1 = f_{m-1}.f_0$.

The output of this CP is

$$P = P_{m-1}\alpha^{m-1} + P_{m-2}\alpha^{m-2} + \ldots\ldots + P_0 = [P_{m-1}, P_{m-2} \ldots\ldots P_0]$$

$$= \begin{cases} A \times B & control = 0 \\ A \times B^2 & control = 1 \end{cases} \quad (6)$$

For the operation, the calculation result can be represented as follows

$$\begin{aligned} P(\alpha) &= P_{m-1}\alpha^{m-1} + P_{m-2}\alpha^{m-2} + \ldots\ldots + P_0 \\ &= A(\alpha) \times B(\alpha) \\ &= A(\alpha).\left(\sum_{j=0}^{m-1} b_j\,\alpha^j\right) \\ &= \sum_{j=0}^{m-1}\left[A(\alpha)\alpha^j\right].b_j \\ &= \sum_{j=0}^{m-1} Q^{j-1}(\alpha).b_j \end{aligned} \quad (7)$$

where $Q^{j-1}(\alpha) = A(\alpha)\alpha^j$. Let $P^{(-1)} = 0$ and $Q^{(-1)} = A(\alpha)$, and then let $P^j(\alpha) = P^{j-1}(\alpha) + Q^{j-1}(\alpha).b^j$ and $Q^j(\alpha) = Q^{j-1}(\alpha).\alpha$ for $0 \le j \le m-1$. Finally, $P(\alpha) = P^{m-1}(\alpha) = A(\alpha) \times B(\alpha)$.

Therein, the computation of $Q^j(\alpha)$ can be performed as follows:

$$\begin{aligned} Q^j(\alpha) &= \sum_{i=0}^{m-1} q_i^{(j)}\,\alpha^i = Q^{j-1}(\alpha).\alpha \\ &= \left(\sum_{i=0}^{m-1} q_i^{(j-1)}.\alpha^i\right).\alpha \\ &= q_{m-1}^{(j-1)}.\alpha^m + \left(\sum_{i=1}^{m-1} q_{i-1}^{(j-1)}.\alpha^i\right) \\ &= q_{m-1}^{(j-1)}.\left(\sum_{i=0}^{m-1} f_i\alpha^i\right) + \left(\sum_{i=1}^{m-1} q_{i-1}^{(j-1)}.\alpha^i\right) \end{aligned} \quad (8)$$

From (8), we obtain the following relation:

$$q_i^{(j)} = q_{m-1}^{(j-1)}.f_i + q_{i-1}^{(j-1)} \quad (9)$$

For the $A \times B^2$ operation, the algorithm can be represented as follows:

$$\begin{aligned} P(\alpha) &= A(\alpha) \times B^2(\alpha) \\ &= A(\alpha).\left(\sum_{j=0}^{m-1} b_j\,\alpha^{2j}\right) \\ &= \sum_{j=0}^{m-1}\left[A(\alpha)\alpha^{2j}\right].b_j \\ P(\alpha) &= \sum_{j=0}^{m-1} Q^{j-1}(\alpha).b_j \end{aligned} \quad (10)$$

where . $Q^{j-1}(\alpha) = A(\alpha)\,\alpha^{2j}$. Let $P^{(-1)} = 0$ and $Q^{(-1)} = A(\alpha)$, and then let $P^j(\alpha) = P^{j-1}(\alpha) + Q^{j-1}(\alpha).b^j$ and $Q^j(\alpha) = Q^{j-1}(\alpha).\alpha^2$ for $0 \le j \le m-1$. Finally, $P(\alpha) = P^{m-1}(\alpha) = A(\alpha) \times B(\alpha)$.

Therein, the computation of $Q^j(\alpha)$ can be performed as follows:

$$\begin{aligned} Q^j(\alpha) &= \sum_{i=0}^{m-1} q_i^{(j)}\,\alpha^i = Q^{j-1}(\alpha).\alpha^2 \\ &= \left(\sum_{i=0}^{m-1} q_i^{(j-1)}.\alpha^i\right).\alpha^2 \end{aligned}$$

$$= q_{m-1}^{(j-1)}.\alpha^{m+1} + q_{m-2}^{(j-1)}.\alpha^m + \left(\sum_{i=2}^{m-1} q_{i-2}^{(j-1)}.\alpha^i\right)$$

$$= q_{m-1}^{(j-1)}.\left(\sum_{i=0}^{m-1} f_i^1\,\alpha^i\right) + q_{m-2}^{(j-1)}.\left(\sum_{i=0}^{m-1} f_i\,\alpha^i\right)$$

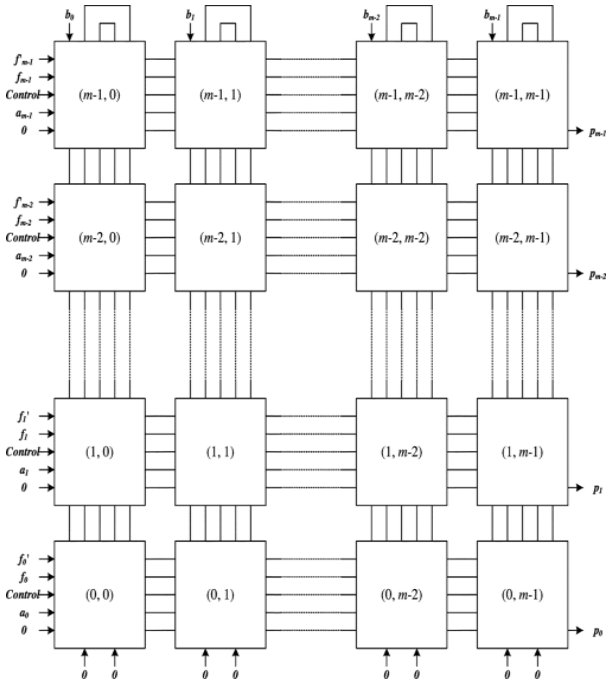


Fig.4.3.1 Structural diagram of CP.

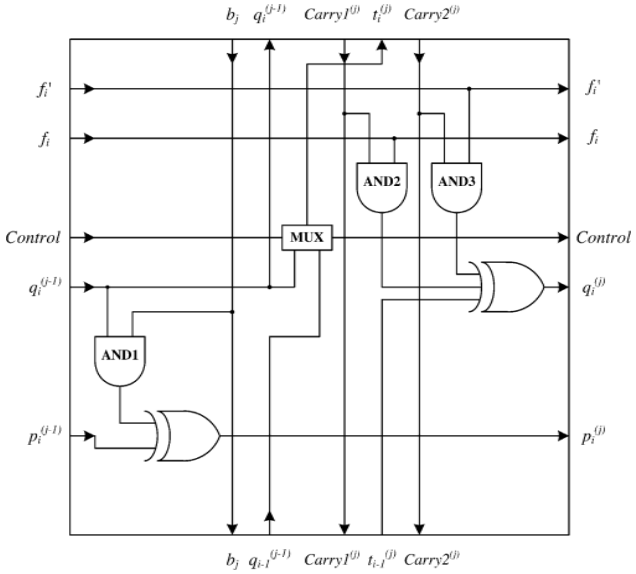


Fig.4.3.2 Circuit of (i,j) identity cell in CP.

$$+\left(\sum_{i=2}^{m-1} q_{i-2}^{(j-1)}.\alpha^{i}\right) \qquad (11)$$

From (11), we obtain the following relation

$$q_i^{(j)} = q_{m-1}^{(j-1)}.\, f_i^1 + q_{m-2}^{(j-1)}.\, f_i + q_{i-2}^{(j-1)} \qquad (12)$$

Figs. 4.3.1 and 4.3.2 show the circuit diagram of the CP and it performs the operation when the control signal control=0.At this time, the input signal $f_i^1 = 0, 0 \le i \le m-1$. As a result,the gate AND3 outputs a 0's, and the multiplexer MUX outputs $t_i^{(j)} = q_i^{(j-1)}$each identity cell. Accordingly, for the (i,j) identity cell

$$Carry1^{(i)} = q_{m-1}^{(j-1)} \qquad (13)$$

$$q_i^{(j)} = Carry1^{(i)}.f_i + t_{i-1}^{(j)}$$

$$= q_{m-1}^{(j-1)}.f_i + q_{i-1}^{(j-1)} \qquad (14)$$

$$p_i^{(j)} = p_i^{(j-1)} + q_i^{(j-1)}.b_j; \qquad (15)$$

Then the m identity cells in the jth column of the CP perform the computations of and . Finally, the output of the CP is P(α)=A(α)×B(α). The CP performs the A× B^2 operation when the control signal control= 1. At this time, the multiplexer MUX of each identity cell outputs $t_i^{(j)} = q_{i-1}^{(j-1)}$.Accordingly

$$Carry1^{(i)} = q_{m-2}^{(j-1)} \qquad (16)$$

$$Carry2^{(i)} = q_{m-1}^{(j-1)} \qquad (17)$$

For the (i,j) identity cell

$$q_i^{(j)} =. Carry1^{(i)} f_i + Carry2^{(i)}.f_i^1 + t_{i-1}^{(j)}$$

$$= q_{m-2}^{(j-1)}.f_i + q_{m-1}^{(j-1)}.f_i^1 + q_{i-2}^{(j-1)} \qquad (18)$$

$$p_i^{(j)} = p_i^{(j-1)} + q_i^{(j-1)}.b_j; \qquad (19)$$

Therefore, the identity cells in the jth column of the CP perform the computations of $Q^j(\alpha)$ and $P^j(\alpha)$. Then the output of the CP is P(α) =A(α)× $B^2(\alpha)$.

Furthermore, to adapt the differentsized finite fields GF(2^m) , each identity cell is further provided with two two-input multiplexers MUX1, MUX2 and a control signal m_i. The control signal m_i is determined by the size of m the finite field GF(2^m) , for controlling the multiplexers MUX1 and MUX2. The control signal $m_{m-1} = 1$ only for the (m-1) th row of identity cells, so that the multiplexer MUX1 can pass $t_{m-1}^{(j)}$to $carry1_i^{(j)}$, , i≤ $m-1$in the same row, and the other multiplexer MUX2 can pass $q_{m-1}^{(j-1)}$ to$carry2_i^{(j)}$, i≤ $m-1$, in the same row. The other control signals $m_i = 1$ for i≠ $m-1$,so that the multiplexer MUX1 in all identity cells for i< $m-1$ can receive $carry1_{i+1}^{(j)} = t_{m-1}^{(j-1)}$ to of the upper identity cell to its $carry2_i^{(j)}$, and the other multiplexer MUX2 in all identity cells for i< $m-1$ can receive $carry2_{i+1}^{(j)} = q_{m-1}^{(j-1)}$ of the upper identity cell to its $carry2_i^{(j)}$. Thus, the m × m identity cells in the lower left part of this general CP perform the same arithmetic operations as the above mentioned m × m CP.Furthermore, the output $p_i^{(M-1)}$,I = m ,m + 1,…….M − 1 are redundancy since input signal $b_j = 0$ for m≤ $j \le M-1$, that is

$$p_i^{(M-1)} = p_i^{(m-1)} + q_i^{(m-1)}.\, b_m$$

$$= p_i^{(m-1)} + q_i^{(m-1)}.\, b_0$$

$$= p_i^{(m-1)}$$

$$p_i^{(m+1)} = p_i^{(m)} + q_i^{(m)}.\, b_{m+1}$$

$$= p_i^{(m)} = p_i^{(m-1)}$$

$$\vdots$$

$$p_i^{(M+1)} = p_i^{(m)} + q_i^{(m)} \cdot b_{m+1} = p_i^{(M-2)} = \cdots = p_i^{(m-1)} . \quad (20)$$

Therefore, the output of the general CP is

$$P = \sum_{i=0}^{m-1} p_i . \alpha^i = \sum_{i=0}^{m-1} p_i^{(M-1)} . \alpha^i = \sum_{i=0}^{m-1} p_i^{(m-1)} . \alpha^i = P^{(m-1)} = \begin{cases} A \times B & control = 0 \\ A \times B^2 & control = 1 \end{cases} \quad (21)$$

Thus, a general CP that can perform AB and AB^2 in differentsized finite field GF(2^m) can be designed.

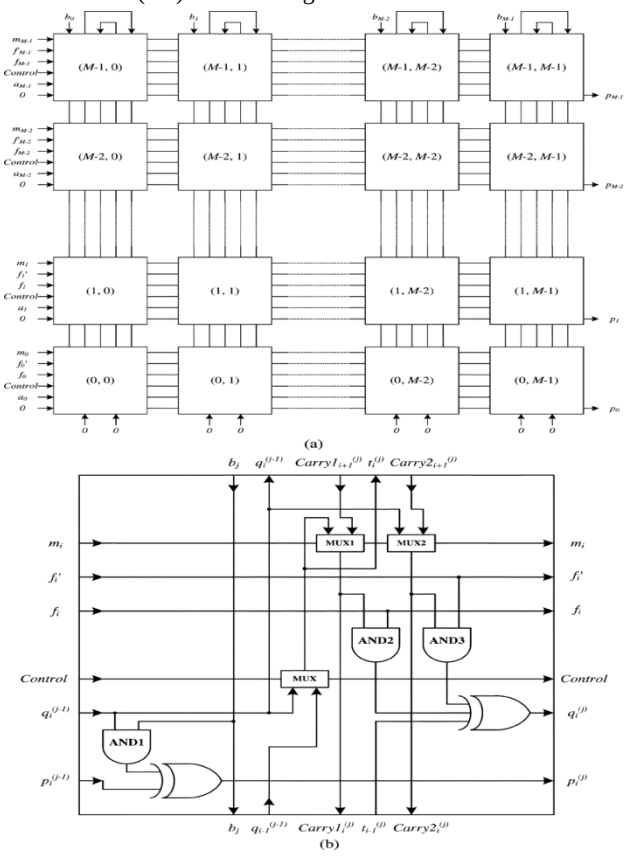


Fig.4.3.3 General CP. (a)Structural diagram. (b)Circuit of (i; j) identity cel

B. Arithmetic Processor (AP)

The AP is structured on the CP, for performing all arithmetic operations except addition. These arithmetic operations can be combined by four basic operations: loading, multiplication, exponentiation, and inverse multiplication. For example, division is implemented by combining multiplication and inverse multiplication. The detailed structural diagram of the AP is shown in the figure.4.3.4.AP which includes a CP and additional control circuits and storage memories. For a finite field GF(2^m) , these control circuits and storage memories includes five m-bit multiplexers, two groups of m -bit D-type flip flops, an m-bit switch and some logic gates generating control signals for multiplexers. the AP can perform all arithmetic operations in the finite field GF(2^m) except addition (accumulation), which can be implemented by the ALU.The input of the AP includes: IN= $(I_{m-1}, I_{m-2}, I_{m-3}, \ldots\ldots I_0)$, control signal , M=$(M_1, M_1 M_\xi), \xi = \lfloor log_2(m-2) \rfloor$ determined by the maximum size of the finite field GF(2m), Signall, Signal2, Controll, Control2, iVm_i, N', Switchl, Switch2 and G .clock, while the output of the AP includes: . OUT =$(O_{m-1}, O_{m-2}, O_{m-3}, \ldots\ldots O_0)$.

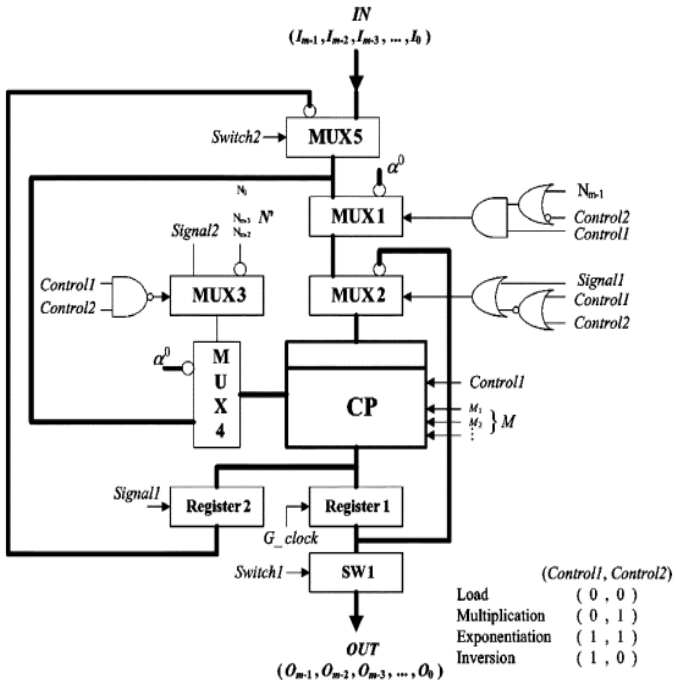


Fig.4.3.4 The structural diagram of AP

Loading: When the control signals (Control1, Control2)=(0,0) , the AP performs the loading operation. This is in order to have the input IN= $(I_{m-1}, I_{m-2}, \ldots\ldots I_0)$ stored in the register Register1 of the AP to serve as an initial value for the next instruction. At this time, the control signals for the multiplexers MUX1-MUX4 are, respectively, 0, 1, 1, 1. If the input IN= $(I_{m-1}, I_{m-2}, \ldots\ldots I_0)$ =β, then two input elements input to the CP are β and α^0. Because the control signal Control1 is 0, the CP performs the A x B operation and the outcome $\beta \cdot \alpha^0$ = βis then loaded to the register Register l.

Multiplication: When the control signals (Control 1, Control2) = (0,1) the AP performs multiplication, multiplying the input IN= $(I_{m-1}, I_{m-2}, \ldots\ldots I_0)$ or the data stored in the register Register2 (determined by the multiplexer MUX5 controlled by the switch signal Switch2) by the data stored in the register Register1. When Switch2 = 1, the AP multiplies the input IN= $(I_{m-1}, I_{m-2}, \ldots\ldots I_0)$ by the data stored in Register l. When Switch2

= 0, the AP multiplies the data stored in Register2 by the data stored in Registerl. When executing this instruction, the CP performs the A x B operation because Control1 = 0, and the control signals of the multiplexers MUX2-MUX4 are, respectively, 0, 1, 1. The outcome is then stored back in Register l.

Exponentiation: When the control signals (Control1, Control2) = (1,1), the AP performs exponentiation (β^N, where $\beta \in GF(2^m)$ and $0 \le N \le 2^m - 1$. Here β is an element in the finite field $GF(2^m)$, which is input from the input IN= $(I_{m-1}, I_{m-2}, \ldots\ldots I_0)$ where N is between 0 and 2^m — 1 and can be divided as $N = N_0 + N_1 .2 + N_2 . 2^2 + \ldots + N_{m-1}.2^{m-1}$, then can be expressed as

$$\beta^N = \beta^{(N_0 + N_1.2 + N_2.2^2 + \cdots + N_{m-1}.2^{m-1})}$$

$$= \beta^{N_0}[\beta^{N_1}[\ldots.\beta^{N_{m-2}}(\beta^{N_{m-1}})^2]^2]^2. \qquad (21)$$

Clearly, exponentiation β^N can be implemented by (m-1)AB^2 operations of the CP. Therefore, the control signal Control1 = 1 for (m - *1*) cycles so that the CP performs the AB^2 operations for (m-1) times. The outcome of the i^{th} cycle is stored in the register Register1 so as to feedback to the CP for the next operation

$$\beta^{N_i} = \begin{cases} \beta & N_i = 1 \\ \propto^0 & N_i = 0 \end{cases} \qquad (22)$$

Furthermore, the outcome of the exponentiation β^{Ni} is selected from α^0 or β according to N_i. The control signal of the multiplexer MUX5 is Switch2 = 1 for (m-1) cycles, the control signal of the multiplexer MUX1 is N_{m-1} for the first cycle, the control signal of the multiplexer MUX2 is Signall = (1, 0, 0,..., 0) for (m - 1) cycles, the control signal of the multiplexer MUX3 is 0 for (m - 1) cycles, the control signal of the multiplexer MUX4 is$N^{'} = (N_{m-2}, N_{m-3}, \ldots\ldots, N_0)$. Thus, the outcome of the exponentiation operation can be obtained in (m - 1) cycles and stored in the register Register1. Moreover, when the exponentiation operation is executing, the outcome for each cycle is stored in the register Register1 Therefore, the data of the previous instruction stored in the register Register1 has to be transferred to the register Register2 (controlled by the signal) for later use.

Inverse Multiplication: When the control signal (Contro1l, Control2) = (1,0), the AP performs inverse multiplicationβ^{-1} where $\beta \in GF(2^{m-1})$, $\beta^{-1}=\beta^{2^m-2}$ Therefore, to perform β^{-1} is to perform the exponention β^N with $N= N_0 + N_1.2 + N_2.2^2 + \cdots + N_{m-1}.2^{m-1}$ clearly, inverse multiplication β^{-1}, like exponentiation β^N, is implemented by (m - 1) AB^2 operations of the CP. Therefore, the control signal Control1 = 1 for (m - 1) cycles, so that the CP performs the AB^2 operation for (m - 1) times. The outcome of the i^{th} cycle is stored in the register Register1 so as to feedback to the CP for the next AB^2 operation. The control signal of the multiplexer MUX5 is Switch2 = 1 for (m - 1) cycles, the control signal of the multiplexer MUX1 is N_{m-1} for the first cycle, the control signal of the multiplexer MUX2 is Signal1 = (1, 0, 0,..., 0) for (m - 1) cycles, the control signal of the multiplexer MUX3 is 1 for (m - 1) cycles, the control signal of the multiplexer MUX4 is Siganl2 = (1,1,1,..., 0). Thus, the outcome of the inverse multiplication operation can be obtained in (m - 1) cycles and stored in Register! Furthermore, when the inverse multiplication operation is executing, the outcome for each cycle is stored in Register1, and so the data of the previous instruction stored in Register1 has to be transferred to Register2 (controlled by the signal Signal1) for later use.Since it is able to perform loading, multiplication, exponentiation and inverse multiplication, the AP can perform all arithmetic operations in the finite field $GF(2^m)$ except addition (accumulation), which can be implemented by the ALU.

C. Arithmetic Logic Unit (ALU)

Addition in the finite field $GF(2^m)$ can be simply implemented by m XOR gates, and another register is provided to store the previous data when performing accumulation. When the accumulation is completed, the register is also refreshed. The overall ALU can be seen in Fig.4.5.1 . This circuit is designed to perform one accumulation in each cycle, which adds the data from the AP and the data stored in the register and outputs back to the register. Whether or not the AP performs accumulation is determined by the control signal

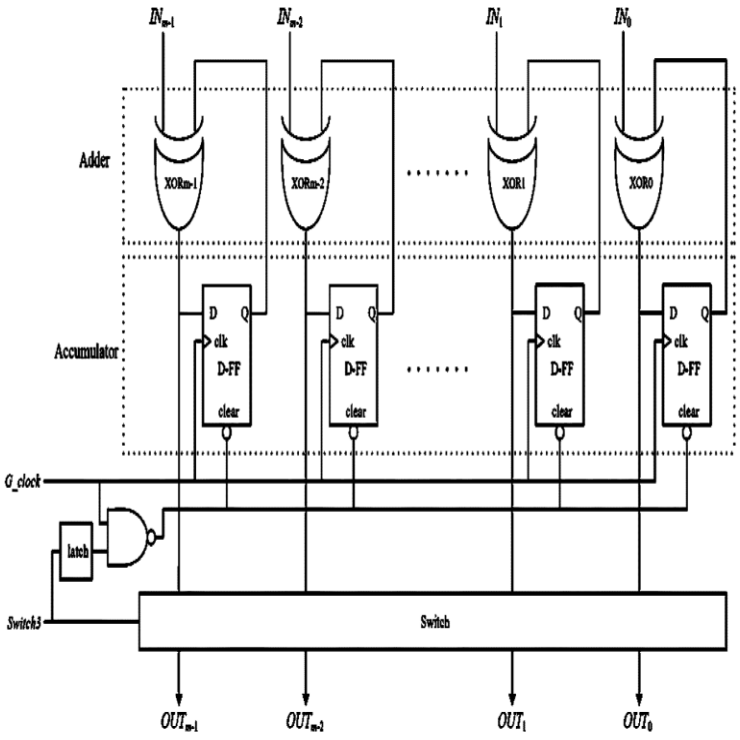
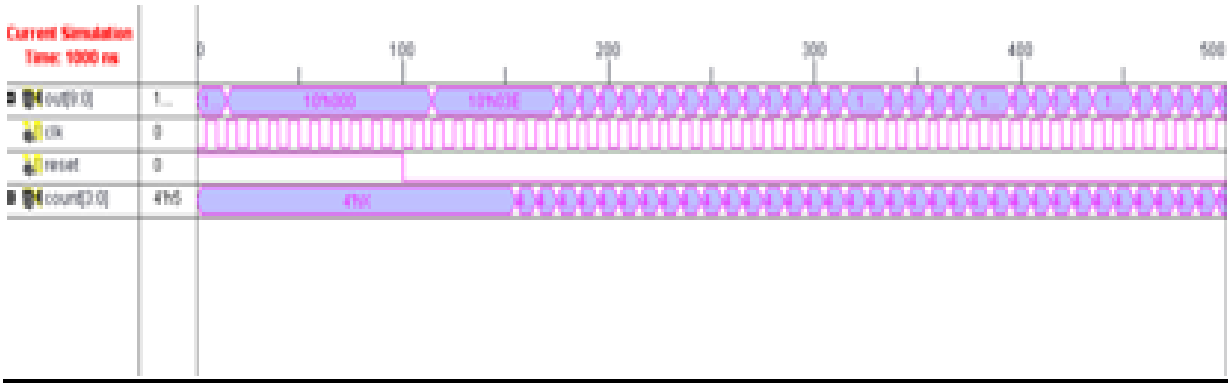


Fig.4.5.1 Circuit diagram of ALU

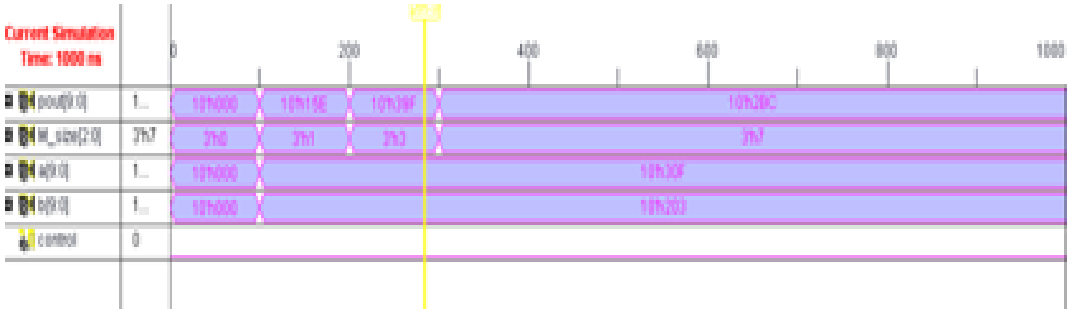
Switch = 1 . When , the ALU receives the output of the AP and performs accumulation. When Switch1 = 0 , a zero element (0) in the finite field $GF(2^m)$ is sent to the ALU, so the output of the ALU remains the same

SIMULATION RESULTS

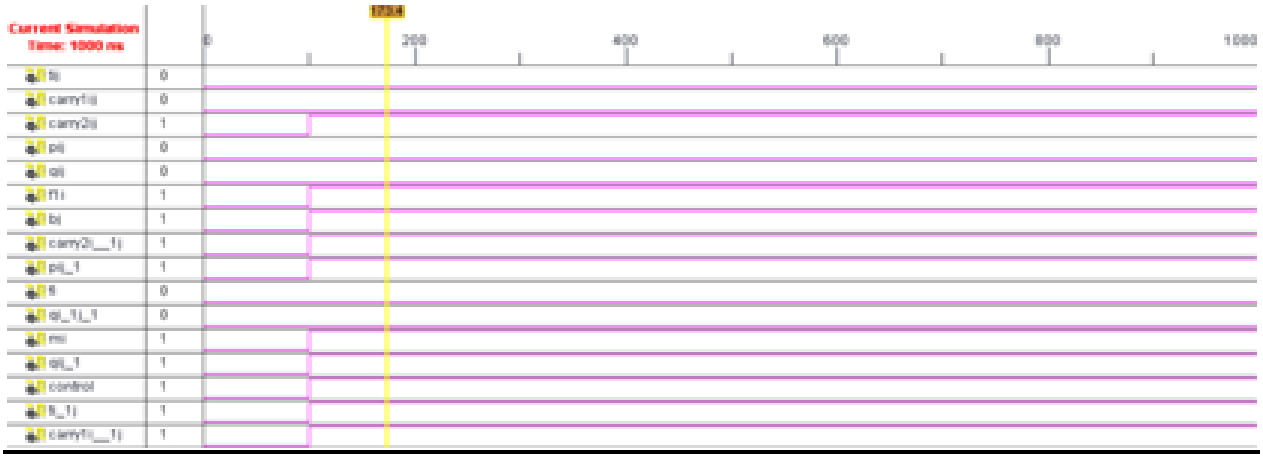
Arithmetic Unit simulation



Calculating_Processor_simulation



Multiplication cell simulation



CONCLUSION

The Finite Field Arithmetic Unit consists of An Arithmetic Processor, An Arithmetic Logic Unit, and A Control unit. The proposed AU has low circuit complexity and is programmable, so that any Error-Correcting decoder that operates in GF(2^m) can be easily implemented with this AU. Herein, the AP is structured by a CP that can perform the A × B and $A \times B^2$ operations in the finite field GF(2^m) .The major job of the presented ALU is to perform additions in the finite field GF(2^m) . By adding the control circuit, all arithmetic operations can be completed using this AU.

ACKNOWLEDGMENT

The authors would also like to thank the references for their valuable comments and some important corrections.

REFERENCES

W. Hoeg and T. Lauterbach , Digital Audio Broadcasting : Principles and Application Digital Radio. Chichester, U.K.: Wiley, 2003.

[2] R. D. Bruin and J. Smits, Digital Video Broadcasting: Technology, Standards, and Regulations. Natick, MA: Artech House, 1999.

[3]C. Smith and J. Meyer, 3GWireless withWiMAX andWiFi: 802.16 and 802.11. NY: McGraw-Hill, 2005.

[4] T. R. N. Rao and E. Fujiwara, Error-Control Coding for Computer B Systems. Upper Saddle River, NJ: Prentice-Hall, 1989.

[5]A. M. Michelson and A. H. Levesque, Error-Control Techniques for Digital
Communication. New York: Wiley, 1985.

[6]S. Lin and D. J. Costellor, Jr., Error Control Coding: Fundamentals and Applications. Englewood Cliffs, NJ: Prentice-Hall, 2004.

[7] S. B.Wicker and V. K. Bhargava, Reed-Solomon Codes and Their Applications.
Piscataway, NJ: IEEE Press, 1994.

[8] B. A. Laws, Jr. and C. K. Rushforth, "A cellular-array multipliers for finite fields GF(2^m)," IEEE Trans. Comput., vol. C-20, no. 12, pp. 1573–1578, Dec. 1971.

[9] C. S. Yeh, I. S. Reed, and T. K. Truong, "Systolic multipliers for finite fields GF(2^m)," IEEE Trans. Comput., vol. C-33, no. 4, pp. 357–360, Apr. 1984.

[10] W. C. Tsai and S. J.Wang, "Two systolic architecture for multiplication in GF(2^m)," Proc. IEEE Comput. Digit. Tech., vol. 147, pp. 375–382, Nov. 2000.

[11] C. H. Kim, C. P. Hong, and S.Kwon, "A digit-serial multiplier for finite field GF(2^m)," IEEE Trans. Very Large Scale Integr. (VLSI) Circuits Syst., vol. 13, no. 4, pp. 476–483, Apr. 2005.

[12] H.Wu, "Bit-parallel finite field multiplier and squarer using polynomial basis," IEEE Trans. Comput., vol. 51, no. 7, pp. 750–758, Jul. 2002.

[13] C. Y. Lee, "Low complexity bit parallel systolic multiplier over GF(2^m) using irreducible trinomials," Proc. IEE Comput. Digit. Tech., vol. 150, no. 1, pp. 39–42, Jan. 2003.

[14] K. Y. Chang, D. Hong, and H. S. Cho, "Low complexity bit-parallel multiplier for GF(2^m) defined by all-one polynomials using redundant representation," IEEE Trans. Comput., vol. 54, no. 12, pp. 1628–1630, Dec. 2005.

[15] C. L. Wang and J. L. Lin, "A systolic architecture for computing inverses and divisions in finite field GF(2^m)," IEEE Trans. Comput., vol. 42, no. 9, pp. 1141–1146, Sep. 1993.

[16] A. V. Dinh, R. J. Bolton, and R. Mason, "A low latency architecture for computing multiplicative inverses and divisions in GF(2^m)," IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process., vol. 48, no. 8, pp. 789–793, Aug. 2001.

O.B.B.Madhuri *,M.Tech VLSI 2'nd Year Student, (Electronics and Communications Engineering), Andhra University/ Sanketika Vidya Parishad Engineering College, visakhapatnam,india,+918143866065.*

E.Rambabu *, Assistant Professor, Electronics and Communications Engineering, Andhra University/Sanketika Vidya Parishad Engineering College, Visakhapatnam, India,+9173827673*

Malijeddi Murali*, Professor & HOD, Electronics and Communications Engineering, Andhra University/Sanketika Vidya Parishad Engineering College, Visakhapatnam, India, +919440149970.,*