Implementation of Polynomial Based Multipliers Using FFA on GF($2^m$)

S. Pushpalatha¹, D. Hari Hara Santosh²
¹M.Tech Student, IEEE Member, ²Assistant Professor, M.V.G.R. College of Engineering, Vizianagaram

Abstract-- This paper describes the implementation of multipliers over finite field with different orders which make use of proposed finite field accumulator (FFA). Using the relation for finite field multiplication, various blocks along with FFA necessary to perform multiplication are derived to form unique multipliers. The relation is reduced in two forms to obtain the two finite field multiplier architectures. First one is bit serial parallel multiplier and Digit serial parallel multiplier. These obtained finite field multipliers are later verified by taking the different orders of m.

Keywords-- FFA; Galois field; Finite field multiplication.

I. INTRODUCTION

Due to the advancement techniques in manufacturing integrated circuits, application to come into picture. Galois field is also known as Finite field. Finite field is defined as field containing finite number of elements. In this paper we solved for binary galois field which is denoted by $2^m$. Where ‘m’ denotes number of digits taken.

Such application is utilized in cryptography to encrypt the data at the transmitter and decrypt the data at receiver of communication system by generating cipher text which consists finite number of elements. To generate the codes in error control coding techniques to test the circuit this galois field is widely used.

On field arithmetic operation varies from off field arithmetic operation. In finite field, addition is performed by simple XOR operation where occurrence of carry propagation is not present. Finite field multiplication is not as simple as addition operation. This operation is quiet complex compared to addition. Finite field operation is performed by modular operation.

There are different basis to represent over GF($2^m$). Those representation are: Normal basis, polynomial basis and last type is dual basis. Normal basis performs by shift and add so more suitable to find inversions while polynomial basis uses a irreducible polynomial given by NIST (National Institute of standards and Technology).

II. FINITE FIELD ACCUMULATOR (FFA)

Consider the operands are A and B, represented by polynomial basis of degree (m-1) as

$$A = \sum_{i=0}^{m-1} a_i z^i$$
$$B = \sum_{i=0}^{m-1} b_i z^i$$  \hspace{1cm} (1)

Addition operation over finite field is performed by bit by bit XOR operations. S=A+B, denoted as

$$S = \sum_{i=0}^{m-1} s_i z^i$$ \hspace{1cm} (2a)

Where $s_i = a_i \oplus b_i$ \hspace{1cm} (2b)

consecutive accumulation for ‘n’ finite field elements $D_j$ where $j$ can be taken as 0,1,2,…...n-1 is denoted by

$$S = \sum_{j=0}^{n-1} D_j$$ \hspace{1cm} (3a)

Where $s_i$ denotes accumulation of $d_{i,0} \oplus d_{i,1} \oplus d_{i,2} \ldots \oplus d_{i,(n-1)}$ \hspace{1cm} (3b)

And $D_j = \sum_{i=0}^{m-1} d_{i,j} z^i$ \hspace{1cm} (3c)

The conventional mean for an FFA over finite field is given away in Fig.1. Every level of bit in accumulation cell composed of XOR gate of two-input as well as D flip-flop. This circuit can be modified in terms of hardware complexity and delay. Using the characteristics table of conventional FFA shown in Fig.2 can lead to proposed FFA that can be designed as shown in Fig.3.
Consider two operands A and B as input and operand C as output over finite field. Then the product of two operands over finite field in polynomial basis is given by

\[ C = B \cdot A \mod Q(z). \]  \hspace{1cm} (4)

Equation (4) can further written as

\[ C = \sum_{i=0}^{m-1} b_i \cdot (z^i \cdot A \mod Q(z)). \]  \hspace{1cm} (5)

Equation (5) represents accumulation over finite field

\[ C = \sum_{j=0}^{m-1} X_j \]  \hspace{1cm} (6)

Where \( X_j \) given by

\[ X_j = b_j \cdot A^j \]  \hspace{1cm} (7)

Where \( A^j \) is given as

\[ A^j = \sum_{i=0}^{m-1} a_i^j \cdot z^i \]  \hspace{1cm} (8)

And \( A^{j+1} = z \cdot A^j \mod Q(z) \)  \hspace{1cm} (9)

Equation (9) represents reduction unit, Equation (7) represents And unit and Equation (6) represents FFA unit. These three units form a serial-parallel architecture for multiplication over finite field shown in Fig.4.

In the above architecture operand A are fed in parallel while operand B fed in serial wise. Reduction cells perform XOR operation if polynomial bit is logic ‘1’ else this unit is removed and data is transferred directly from one D flip flop to other. For first cycle operand A loads the D flip-flops of reduction unit. For every cycle, modular reduction unit (MRU) performs XOR operation if coefficient value ‘i’ of polynomial is ‘1’ as shown in (12). Later in AND unit operand B feeds the data to perform and operation with the data obtained at the output of MRU according to (7). At last accumulation of obtained result is done according to (6).

Consider \( m = 8 \), then irreducible polynomial is \( x^8 + x^4 + x^3 + x^2 + 1 \) that can be denoted as 10001101 in binary form.

\[ a_0^{i+1} = a_{m-1}^i \]  \hspace{1cm} (10)

\[ a_j^{i+1} = a_j^i \text{ for } j=1,5,6,7 \]  \hspace{1cm} (11)
\[ a_j^{i+1} = a_{j-1}^i + a_j^i \text{ for } j=2,3,4 \]  \hspace{1cm} (12)

\[ a_2^{i+1} = a_1^i + a_2^i \]  \hspace{1cm} (13)

\[ a_3^{i+1} = a_2^i + a_3^i \]  \hspace{1cm} (14)

\[ a_4^{i+1} = a_3^i + a_4^i \]  \hspace{1cm} (15)

The MSB bit shifts to LSB for next cycle as from (10). Coefficient value is ‘0’ then forms (11) otherwise forms (12) if coefficient value is 1’. This is as shown in Fig.5.

The architecture consists of three units: Modular reduction unit, AND section and addition section as in (21),(18) and (19).

\[ z^i . A^j \mod Q(z) = \sum_{k=0}^{m-i-1} a_k^j z^{ki} \mod Q(z) \]  \hspace{1cm} (20)

\[ = \sum_{k=0}^{m-i-1} a_k^j z^{ki} + \sum_{k=0}^{m-i-1} a_{(m+i+k)}^j z^k \mod Q(z) \]  \hspace{1cm} (21)

Fig6: Digit-serial-parallel multiplier over GF (2^m)

Input register consists D Flip-flops are loaded by operand A. The reduction section performs XOR operation with polynomial coefficient. AND section performs and operation with operand B. Addition section performs addition of XOR gates and are accumulated in FFA.

\[ c = e^{\log(a \cdot b)} \]  \hspace{1cm} (22)

A. Standard polynomials as per NIST

For m=4, Polynomial is \( x^4 + x^3 + 1 \), its binary form is 10010. For m=8, Polynomial is \( x^8 + x^4 + x^3 + x^2 + 1 \), its binary form is 10011101. For m=16, Polynomial is \( x^{16} + x^4 + x^3 + x + 1 \), its binary form is 100000000000101011
B. Simulation Results

Answers can be verified using the logarithmic and exponential tables. Let us verify this for 8 bits.

Irreducible Polynomial for 8 bits is $X^8+X^4+X^3+X^2+1$.

Let $A = 8\text{'h}83$ and $B = 8\text{'h}57$

57 * 83 = C1 = 01010111 * 10000011 = 11000001

**Step 1:** From the logarithm table find $L(57)$ and $L(83)$.

$L(57) = 62$ and $L(83) = 50$.

**Step 2:** Add them together (regular addition) to get $B2$.

\[
\begin{array}{c}
01100010 \\
01010000 \\
\hline \\
10110010
\end{array}
\]

Let $B2 = 10110010$.

**Step 3:** Next, take the exponential of $B2$ to get the product $C1$.

$E(B2) = C1$

Therefore $57 * 83 = C1$.

Below are the logarithmic and exponential tables to check the theoretical calculations with those of practical ones.
Table 1
Logarithms over GF($2^m$)

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-</td>
<td>00</td>
<td>10</td>
<td>20</td>
<td>30</td>
<td>40</td>
<td>50</td>
<td>60</td>
<td>70</td>
<td>80</td>
<td>90</td>
<td>a0</td>
<td>b0</td>
<td>c0</td>
<td>d0</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>e</td>
<td>f</td>
<td>a</td>
<td>b</td>
<td>c</td>
<td>d</td>
<td>e</td>
</tr>
</tbody>
</table>

Table 2
Exponentials over GF($2^m$)

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>01</td>
<td>03</td>
<td>05</td>
<td>0f</td>
<td>11</td>
<td>33</td>
<td>55</td>
<td>0f</td>
<td>11</td>
<td>33</td>
<td>55</td>
<td>0f</td>
<td>11</td>
<td>33</td>
<td>55</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>e</td>
<td>f</td>
<td>a</td>
<td>b</td>
<td>c</td>
<td>d</td>
<td>e</td>
</tr>
</tbody>
</table>

103
Acknowledgement

Our sincere thanks to Dr. R. Ramana Reddy, Professor and Head of the Department for M.V.G.R College of engineering for his support in bringing this paper and also like to express my thanks to all the staff of the M.V.G.R College in making this paper.

REFERENCES


