Implementation of reconfigurable galois field multipliers over $2^m$ using primitive polynomials

B. Raj Narain 1*, Dr. T. Sasilatha 2

1 Research scholar, Anna University, Chennai, India
2 Dean, Department of Electrical and Electronics (Marine), AMET university, Chennai
*Corresponding author E-mail: rajnarainb@gmail.com

Abstract

The Galois field multiplier finds extensive use in cryptographic solutions and applications. The Galois field multiplier can be implemented as fixed bitwise or reconfigurable. For fixed length, the data is restricted to the fixed length. But in reconfigurable GF multipliers, the bit length of the multiplier is flexible and is independent of hardware architecture. This paper proposes a method to implement a reconfigurable GF multiplier for various bit values from 8 to 128 bits. This paper compares the area complexity of various bit size in Xilinx Spartan 3E family FPGA and estimates the resources required for the implementation.

Keywords: Galois Field; Reconfigurable; Primitive Polynomial

1. Introduction

The study of Galois Field which was entitled later as Evariste Galois, and also recognised as finite field, denotes to an arena in which there happens finitely various components. It is predominantly expedient in interpreting computer data as they are characterised in binary logic. That is, computer data comprise of blend of two numbers, 0 and 1, which are the modules in Galois field whose number of components is two. Demonstrating data as a vector in a Galois Field permits mathematical procedures to ascend data effortlessly and efficiently.

This Galois Field (GF) is extensively castoff in digital signal processing, cryptography and channel coding. There have been various works on planning cost effective encryption hardware used in battery based applications. Most work emphases on area reduction and propagation delay or critical path. In [1], the writersadvise a low cost technique to device limited 8 bit multipliers. In [2], a completely rolled pipelined architecture on basis exchange un it was focused.

A technique consuming composite field arithmetic to moderate hardware complexity in modular transposal above GF($2^m$) was planned in [3]. In [4], a scalable multiplier involving of several processing components connected in a parallel manner was planned, where each PE comprises two w-bit Carry save adders. Its performance is reliant on tractability of performance/area trade-off in numerous applications. There are two bases used to represent the finite field elements over GF ($p^n$), namely normal basis and polynomial basis.

2. Previous works

In the collected works, numerous algorithms and hardware architectures for the regular source multiplication over GF ($2^m$) are offered. The hardware execution of finite field multipliers expending normal basis can be categorized into three strategies.

The first technique is established on bit level execution [5]–[8]. In [5], Massey and Omura conceived a bit-level GF ($2^m$) standard basis multiplier. In that circumstance, a bit-level multiplier receipts $m$ clock cycles to compute one multiplication above a binary field of size $m$. In [6], Gao et al. offered developments over the Massey-Omura multiplier and condensed the area and power depletion. In [7], Beth et al. offered approaches for VLSI execution of public key algorithms and finite field arithmetic. In [8], Agnew et al. offered an execution of the exponentiation in GF ($2^m$), where its foremost disadvantage is the implementation period to achieve the multiplication for huge field proportions. Nearly some of the workings [9] are executed in bit serial or bit parallel to condense area intracacies reliant on the hardware.

The secondary one is grounded on a parallel-level execution [9]–[12]. In [9], [11] and [12], it is instigated a similarform of the Massey-Omura multiplier by confiscating the dismissal. In [9], Sunar and Kocplanned a regular basis multiplier created on the canonical basis multiplier. In [9], their period complexity is a lesser amount of than those produced by All One Polynomials (AOP), and the quantity of XOR and AND gates is aroundpartial than that of the Massey-Omura multiplier over GF ($2^m$). The bit serial multiplication erected on polynomial basis was scheduled in [20] [21]. Bit parallel multipliers are completed in [22], [23] and [26].

The bit-serial multipliers prompt the outcome of two ‘m’ bit operands subsequently ‘m’ clock cycles (latency) [27] and [28]. The bit-serial multipliers deal a benefit of reduced hardware means of intakes. The detriment is the greater quantity of essential clock cycles (latency) which consequences lower presentation. Owing to the less significant hardware operation, bit-serial multipliers are often applied for inhibited requests, where low extent is the important pre-requisite.
3. Galois fields

A field is a customary of elements with two practicedemarcated arithmetic procedures: furthermoreadditionally, addition and multiplication. Thus the elements of the field are called as an additive abelian assembly, and the non-zero elements of the field are called as a multiplicative abelian assembly. These resources that all elements of the field have an additive inverse and all non-zero elements have a multiplicative inverse. A field is also called finite if it certainly has a finite quantity of elements.

The elements of Galois Field GF (p^n) is defined as

\[ GF(p^n) = \{ 0, 1, 2, \ldots, p^n-1 \} \]

\[ U(p, p+1, p+2, \ldots, p^n-1) \]

Where the \( p \in \mathbb{P} \) and \( n \in \mathbb{Z}^+ \). The order of the field is given by \( p^n \) while \( p \) is called the characteristic of the field. On the other hand, \( p^n \) as one may have guessed it, stands for Galois Field. Also note that the degree of polynomial of each element is at most \( n-1 \).

4. Galois Field Multiplication

A circuit effects regular multiplication of two field elements in a Galois field GF (2^m). Each of the field elements is expressed by an m-bit binary number. The two field elements are applied to a binary multiplier array which generates 2m-bit partial products. The partial products are divided by a generator polynomial of the Galois field to produce final m-bit binary products.

Let \( a(x) \) and \( b(x) \) be two field elements and \( s(x) \) be their product. Then,

\[ s(x) = a(x)b(x) \mod p(x) \] (1)

Thus, the polynomial basis multiplication involves two steps: polynomial multiplication and reduction modulo an irreducible polynomial as shown in figure 1 for a 4 bit multiplier. The product \( d(x) \) of the polynomials representing the field elements \( a(x) \) and \( b(x) \), \( d(x) = a(x)b(x) \), is a degree 2m – 2 polynomial. In the modular reduction \( s(x) = d(x) \mod p(x) \), the degree 2m – 2 polynomial \( d(x) \) is reduced by the degree m irreducible polynomial \( p(x) \) iteratively. The choice of the irreducible polynomial \( p(x) \) may ease the modular reduction. Sparse irreducible polynomials having fewer nonzero terms are usually preferred for efficiency.

Example for Galois field Multiplication:

If \( A = x^1 + x^2 + 1 \) (1101 in binary) and \( B = x^2 + x \) (0110 in binary) are two polynomials, then \( A \times B \) is called polynomial multiplication which returns \( x^3 + x^2 + x^2 + x \), if \( m = 4 \). The result should be reduced to a degree less than 4 by irreducible polynomial \( x^4 + x + 1 \).

Therefore

\[ A \times B \mod P = x^5 + x^3 + x^2 + x \mod p(x) \]

\[ = (x^4 + x + 1) x + x^5 + x^4 + x^2 + x \]

\[ = 2x^5 + x^3 + 2x^2 + 2x \]

\[ = x^5 \mod 2 \]

For implementing the modular multiplication the following irreducible or primitive polynomials are considered.

<table>
<thead>
<tr>
<th>Table 1: Primitive Polynomials</th>
</tr>
</thead>
<tbody>
<tr>
<td>( m )</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>8</td>
</tr>
</tbody>
</table>

This paper proposes that Reconfigurable Galois field multiplier which is applicable at for length GF multiplication by changing primitive polynomial for different GF. Actually, the multiplier consists of AND gates and special cells as shown in figure 2, which consist of product and reduction. The distinct cell estimates partial product and partial reduction at the meantime which is similar to binary product and division according to \( a(x) \cdot b(x) \mod p(x) \). Distinct cell that receives 2 multipliers, precedent result (pr), primitive polynomial (pp), quotient (q), and produces an output

\[ S = a \cdot b \oplus p \cdot p \cdot p \cdot p \] (2)

For GF (2^m) the multiplier cell consists of 4 AND gates and 12 special cells, which has rows similar to Linear Polynomial Reduction and the dotted line is quotient to use in the next row. Distinct cells are according to a relationship of wiring, for example equation 2 can be written as below:

\[ s_{ij} = a_{m-j} \cdot b_{m-i} \oplus p_{m-j} \cdot S_{i+1,j} \oplus S_{i,j+1} \] (3)
4. Reconfigurable Galois Field multipliers

In order to randomly calculate different Galois Field multiplication, one of the methods is to include an additional Galois Field length controller. The Galois Field length controller makes GF multiplier architecture to receive different irreducible polynomials. The GF length controller shown in figure 3 consists of memory and multiplexer which receives a control signal and gives the primitive polynomial. The memory of GF length controller initialises the primitive polynomials and connects multiplexer to choose another primitive polynomial. The control signal is the selecting of multiplexer which connects the input to the output. Galois Field multiplier arrangement modifies the data path allowing to primitive polynomial. The user adds the primitive polynomial to the memory rendering to the need of GF length, while the Galois Field multiplier arrangement covers the largest Galois Field length. For example when the length of the Galois Field is m = 8 it can determine any Galois Field multiplication for m ≤ 8. The memory initialises primitive polynomial for m=4 to m=8 and The Galois Field multiplier structure expands to 8x8.

5. Simulation result

The multiplier has been implemented for 4 and 8 bits for the polynomial $1+x^2+x^3+x^4+x^8$. Even though various polynomials are used for reducing the result of the Galois field multiplication, the output result may vary but is limited to the m value. The multipliers are implemented in the Xilinx IDE for the devices Spartan 3E. The results shown in figure 4 and 5 were obtained using Xilinx ISE tool by synthesising for the Spartan 3E family’s device XC3S500E. The synthesis results in table 2 shows the number of Lookuptables and slices required for applying the 8, 16, 32 and 64 bit multipliers over Galois Field of $2^m$. The table 3 compares the number of slices required to implement 8 and 16 bit multipliers of this work with [25] and [26]. Thus we can observe that the number of look up tables and slices required for different m values are increasing.

6. Conclusion

The synthesis is completed for 8 to 64 bits using the same algorithm. The power is also estimated at an ambient temperature of 25°C. It is observed that if the temperature of the atmosphere increases, the leakage current and subsequently the power consumed also increases. In addition to that, due to the absence of clock input in the synthesis, the delay occurring due to clock input is reduced considerably. Since the output is unique for different polynomials, it can be used as a key in cryptographic applications without much change in the hardware. The reconfigurable Galois field multiplier
can also be replaced with other methods such as Karatsuba – Ofman algorithm, Montgomery multiplier for further reduced hardware and low power consumption when more number of bits is used.

References


