MEMO: GFAU System Design Document
DATE: May 8, 2018
TO: Dr. E.F. Charles LaBerge
SUBJECT: Mathematical Background on the Galois Field Arithmetic Unit

1 Background

Galois fields (denoted $GF(p^n)$, where $p, n \in \mathbb{N}_{>0}$, and p is prime) are fields with finite orders of p^n. They are a key part of number theory, abstract algebra, arithmetic algebraic geometry, and cryptography. In error detection and correction, Galois fields are utilized in cyclic redundancy check (CRC) which are used in digital networks and storage devices to detect accidental changes to raw data.

The Galois Field Arithmetic Unit (GFAU) is a scalable arithmetic logic unit (ALU) capable of generating elements in the Galois field of an irreducible polynomial in $GF(2^n)$, where $2 \leq n <$ (# of memory address pins). GFAU supports addition, subtraction, multiplication, division and logarithm of these elements for low powered devices.

1.1 Purpose and Scope

This document will demonstrate the functionality of the unit by deriving all its functionality through mathematical approaches. The mathematical algorithms will be followed by their corresponding implementations in digital design. The algorithms will be followed by brief documentation on the modules included in the unit.

1.2 Terms and Keywords

1.2.1 Input Primitive Polynomials

Input primitive polynomials in the Galois Field are represented as:

$$c_n x^n + \ldots + c_2 x^2 + c_1 x^1 + c_0 x^0, \text{ where } c, x \in GF(2) = \{0, 1\}$$

For convenience and simplicity, all the examples provided will refer to the polynomial: $x^3 + x^2 + x^0$.

1.2.2 Elements

The elements of an input polynomial refer to the $2^n - 1$ elements in the field.

1.2.3 Polynomial Form

The polynomial forms of the elements refer to the $2^n - 1$ symbolic representations of the input primitive polynomials in the field.

1.2.4 Example

An example of the elements and their corresponding polynomials is provided below:

Table 1: The 8 Element Vectors of $x^3 + x^2 + x^0$ in $GF(2)[x]$

Element	**Symbol**	**Polynomial Form**	**Symbol**
0 (NULL)	[1]	$0+0+0$	000
β^0	000	$0+0+\beta^0$	001
β^1	001	$0+\beta^1+0$	010
β^2	010	β^2+0+0	100
β^3	011	$\beta^2+0+\beta^0$	101
β^4	100	$\beta^2+\beta^1+\beta^0$	111
β^5	101	$0+\beta^1+\beta^0$	011
β^6	110	$\beta^2+\beta^1+0$	110
β^7	[2]	$0+0+\beta^0$	001

[1] The additive identity, 0 (zero), referred to as NULL, in its element form is reserved where its binary symbol does not represent its decimal value.
[2] Elements beyond the $(2^n - 1)$th element will be handled with special conditions since they cycle back to previous elements.

1.2.5 Notations

Several notations are used in this document to assist in linking the mathematical and digital design concepts. A list of selected notations are provided below:

Table 2: Mathematical and Logical Notations

Notation	**Definition**
$a + b$	Arithmetic addition of a and b
$a - b$	Arithmetic subtraction of a and b
$a \cdot b$	Arithmetic multiplication of a and b
$a \,/\, b$	Arithmetic Division of a and b
$a \wedge b$	Logical conjunction of a and b
$a \vee b$	Logical disjunction of a and b
$a \oplus b$	Logical exclusive disjunction of a and b
$\overline{a}$	Logical complement of a
$a \ll b$	Logical shift-left a by b bits
$a_{2's}$	Two's complement of a
$\|a\|$	Number of bits in a
$< x_n, x_{n-1}, \ldots, x_0 >$	Ordered array of size n
$\{x_n, x_{n-1}, \ldots, x_0\}$	Unordered set of size n
$< x_m, \overleftarrow{0_{m-1}, \ldots, 0_n}, x_{n-1}, \ldots, x_0 >$	Zero-padding between the $(m-1)$th and nth bits
$A[i]$	ith bit of array A
$A[i:j]$	Subarray of array A from its ith to jth index inclusive; where $i > j$

2 Algorithms

Input polynomials will be represented as n-bit binary-coded decimal (BCD) arrays.

For example, the polynomial $x^3 + x^2 + x^0$ will be represented as

$$< 0000\ 0000\ 0000\ 1101 > \text{(3rd, 2nd, and 0th bits)}$$

2.1 Primitive Polynomial

A polynomial is said to be irreducible if and only if there exists no roots for it. A primitive polynomial is an irreducible polynomial that generates all elements of an extension field from a base field. [1]

2.2 Symbols

Once a polynomial is determined irreducible, its symbols may be generated. The number of terms grow exponentially, $2^n - 1$, where n is the highest degree of the polynomial.

Table 3: Generated Symbols for Irreducible Polynomials of Degree n in Memory with $m > n$ Address Pins

Element	Polynomial Form	Symbol
0	$0_m + \ldots + 0_n + \ldots + 0_2 + 0_0 + 0_0$	$< \overleftarrow{0_m \ldots 0_n} \ldots 0_2 0_1 0_0 >$
α^0	$0_m + \ldots + 0_n + \ldots + 0_2 + 0_0 + \alpha_0^0$	$< \overleftarrow{0_m \ldots 0_n} \ldots 0_2 0_1 1_0 >$
α^1	$0_m + \ldots + 0_n + \ldots + 0_2 + \alpha_1^1 + 0_0$	$< \overleftarrow{0_m \ldots 0_n} \ldots 0_2 1_1 0_0 >$
α^2	$0_m + \ldots + 0_n + \ldots + \alpha_2^2 + 0_1 + 0_0$	$< \overleftarrow{0_m \ldots 0_n} \ldots 1_2 0_1 0_0 >$
$\ldots$	$\ldots$	$\ldots$
α^{n-1}	$0_m + \ldots + \alpha_{n-1}^{n-1} + \ldots + 0_2 + 0_1 + 0_0$	$< 0_m \ldots 1_{n-1} \ldots 0_2 0_1 0_0 >$
α^n	$0_m + \ldots + \alpha_{n-1}^{n-1} + \ldots + \alpha_2^2 + \alpha_1^1 + \alpha_0^0$	$< 0_m \ldots x_{n-1} \ldots x_2 x_1 x_0 >$
α^{2^n-1}	$0_m + \ldots + 0_n + \ldots + 0_2 + 0_0 + \alpha_0^0$	$< \overleftarrow{0_m \ldots 0_n} \ldots 0_2 0_1 1_0 >$

2.2.0.1 Digital Logic

Generating the rest of the symbols may be implemented using the following recurrence relation:

$$\begin{aligned}\alpha^{n+m} &= \alpha^{n+(m-1)} \times \alpha^{n} \\ &= \begin{cases} \alpha^{n+(m-1)} \ll 1 & \text{if } \alpha^{n+(m-1)}[n-1] = 0 \\ (\alpha^{n+(m-1)} \ll 1) \oplus \alpha^{n} & \text{if } \alpha^{n+(m-1)}[n-1] = 1 \end{cases}\end{aligned}$$

2.3 Operations

The Galois Field Arithmetic Unit supports addition, subtraction, multiplication, division and logarithm of Galois terms.

2.3.1 Addition and Subtraction

Binary addition and binary subtraction are synonymous in the Galois Field. Addition and subtraction of Galois operands may be done by computing the bitwise exclusive disjunction of the operands.

$$\begin{aligned}\alpha^{i} \pm \alpha^{j} &= \{x_{i,n}, \ldots x_{i,2}, x_{i,1}, x_{i,0}\} \pm \{x_{j,n}, \ldots x_{j,2}, x_{j,1}, x_{j,0}\} \\ &= \{(x_{i,n} \oplus x_{j,n}), \ldots, (x_{i,2} \oplus x_{j,2}), (x_{i,1} \oplus x_{j,1}), (x_{i,0} \oplus x_{j,0})\} \\ &= \alpha^{k}\end{aligned}$$

2.3.1.1 Digital Design

Only polynomial forms of inputs are valid for these operations.

The implementation of Galois addition and subtraction may be computed with a single-level parallel array of XOR gates.

2.3.2 Multiplication

Binary multiplication of Galois operands is congruent to the sum of the indices of the operands. If the indices sum to greater than or equal to $2^n - 1$, then $2^n - 1$ is subtracted from the sum to prevent overflow.

$$\begin{aligned}
\alpha^i \cdot \alpha^j &= \{x_{i,n-1}, \dots, x_{i,2}, x_{i,1}, x_{i,0}\} \cdot \{x_{j,n-1}, \dots, x_{j,2}, x_{j,1}, x_{j,0}\} \\
&= \alpha^{(i+j) \pmod{(2^n-1)}} \\
&= \begin{cases} \alpha^{(i+j)-(2^n-1)} & \text{if } (i+j) \geq 2^n - 1 \\ \alpha^{(i+j)} & \text{if } (i+j) < 2^n - 1 \end{cases}
\end{aligned}$$

2.3.2.1 Digital Design

Only element forms of inputs are valid for this operation.

Galois multiplication requires multiple binary additions and condition checks. To find the product of α^i and α^j, a carry-lookahead adder may be used to sum the elements $i + j$. The most significant bits of the sum and the sum $+1$ will then be OR-ed to compute a single-bit control signal. The control signal will be multiplexed with a binary 1 to be added to the sum.

$$\begin{aligned}
\alpha^i \cdot \alpha^j &= \{x_{i,n-1}, \dots, x_{i,2}, x_{i,1}, x_{i,0}\} \cdot \{x_{j,n-1}, \dots, x_{j,2}, x_{j,1}, x_{j,0}\} \\
&= \begin{cases} \alpha^{(i+j)} & \text{if } \Big((i+j)[n] \ \vee \ (i+j+1)[n]\Big) = 0 \\ \alpha^{(i+j+1)} & \text{if } \Big((i+j)[n] \ \vee \ (i+j+1)[n]\Big) = 1 \end{cases}
\end{aligned}$$

2.3.3 Division

Binary division of Galois operands is congruent to the difference of the indices of the operands. If the difference is negative, then the absolute value of the difference is subtracted from $2^n - 1$ to prevent underflow. If the difference is zero, then the quotient is α^0.

$$\begin{aligned}
\alpha^i/\alpha^j &= \{x_{i,n-1},\ldots x_{i,2},x_{i,1},x_{i,0}\}/\{x_{j,n-1},\ldots x_{j,2},x_{j,1},x_{j,0}\} \\
&= \alpha^{(i-j) \pmod{(2^n-1)}} \\
&= \begin{cases} \alpha^{(2^n-1)-(j-i)} & \text{if } (i-j)<0 \\ \alpha^{(i-j)} & \text{if } (i-j)>0 \\ \alpha^0 & \text{if } i=j,\ i\neq 0 \\ ERROR & \text{if } j=0 \end{cases}
\end{aligned}$$

2.3.3.1 Digital Design

Only element forms of inputs are valid for this operation.

Galois division requires multiple binary additions and condition checks. To find the quotient of α^i and α^j, the two's complement of j has to first be summed with i. Converting to two's complement may be done with parallel inverters and a carry-lookahead adder. The overflow bit of the sum will then be used as a control signal to a multiplexer. If activated, the multiplexer will add the two's complement of a binary 1 to the sum.

$$\begin{aligned}
\alpha^i/\alpha^j &= \{x_{i,n-1},\ldots,x_{i,2},x_{i,1},x_{i,0}\}/\{x_{j,n-1},\ldots,x_{j,2},x_{j,1},x_{j,0}\} \\
&= \begin{cases} \alpha^{(i+j_{2's})} & \text{if } (i+j_{2's})[n]=1 \\ \alpha^{(i+j_{2's}+1_{2's})} & \text{if } (i+j_{2's})[n]=0 \end{cases}
\end{aligned}$$

2.3.4 Logarithm

Logarithm is considered a unary operation in the Galois Field, where only one operand is required. The base of the logarithm operation in the Galois Field is implicitly 2. The logarithm of a Galois operand is congruent to its index.

$$log_{GF(2)}(\alpha^i) = i$$

2.3.4.1 Digital Design

Only element forms of inputs are valid for this operation.

The logarithm of a Galois operand is the index of the term itself.

2.4 Arithmetic Exceptions

2.4.1 Zero

Zero, or the null symbol, is the numerical zero used to fulfill its role as the additive identity in the Galois field. The following operations demonstrate the characteristics of zero in the Galois field.

Table 4: Operations with $\mathbf{0}$ in $GF(2)[x]$

Operand 1	**Operation**	**Operand 2**	**Output**
$\mathbf{0}$	$+/-$	β^x	β^x
β^x	$+/-$	$\mathbf{0}$	β^x
$\mathbf{0}$	$\times$	β^x	$\mathbf{0}$
β^x	$\times$	$\mathbf{0}$	$\mathbf{0}$
$\mathbf{0}$	$\div$	β^x	$\mathbf{0}$
β^x	$\div$	$\mathbf{0}$	$ERROR$
$\mathbf{0}$	deg	$DON'TCARE$	$ERROR$

2.4.1.1 Digital Design

To handle zero, multiplexers are used to decode the intended operation to determine the forms if the inputs. Once the form is determined, the module raises a `zero flag` which is handled by individual operations to raise exceptions.

$$\texttt{zero flag} = \begin{cases} \wedge\big(\text{operand}\big), & \text{if operation} \in \{\text{addition, subtraction}\} \\ \vee\big(\text{operand}\big), & \text{if operation} \in \{\text{multiplication, division, logarithm}\} \end{cases}$$

2.4.2 Membership

References

[1] E. W. Weisstein. Primitive polynomial. *MathWorld – A Wolfram Web Resource*, Dec 2017.