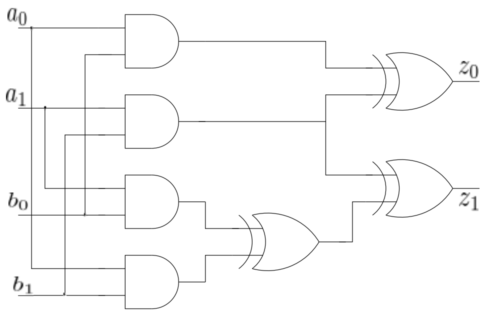
Hardware Datapath Verification using Algebraic Geometry

Priyank Kalla



$$\text{ideal } J = \begin{cases} A = a_0 + a_1\alpha \\ B = b_0 + b_1\alpha \\ Z = z_0 + z_1\alpha \\ z_0 = a_0b_0 + a_1b_1 \\ z_1 = a_0b_1 + a_1b_0 + a_1b_1 \end{cases}$$

Is $Z = AB \in I(V(J))$?

http://www.ece.utah.edu/~kalla

Notation

$\mathbb{N}$	natural numbers $\{0, 1, 2, \dots\}$
$\mathbb{Z}$, $\mathbb{Z}_n$	ring of integers, integers $\pmod n$
$\mathbb{F}$	any field
$\mathbb{R}, \mathbb{Q}, \mathbb{C}$	reals, fractions, complex numbers
$\mathbb{F}_q$	finite (Galois) field of q elements
$\mathbb{B}$, $\mathbb{B}^k$	Boolean, k-dimensional Boolean space
$\mathbb{F}_{2^k}$	the Galois field of 2^k elements
$\mathbb{Z}_{2^k}$	finite integer ring $\pmod{2^k}$
$\overline{\mathbb{F}}$	the algebraic closure of $\mathbb{F}$
$f : A \to B$	function mapping from A to B
$\mathbb{F}[x_1, \dots, x_n]$	multi-variate polynomial ring with coefficients in $\mathbb{F}$
$J = \langle f_1, \dots, f_s \rangle$	ideal generated by polynomials
$V_{\mathbb{F}}(J)$	variety of ideal J over the field $\mathbb{F}$
$I(V(J))$	ideal of polynomials that vanish on $V(J)$
J_0	ideal of polynomials that vanish on all inputs
$\mathcal{M}_{n\times m}$	$n \times m$ matrices
$V \cong W$	isomorphic spaces
$Z_{n\times m}$ or Z, $I_{n\times n}$ or I	zero matrix, identity matrix
$\lvert T \rvert$	determinant of the matrix

Preface

As of October 29, 2014, this is book is a work in progress. Likewise, this Preface a work in progress too.....

I am writing this book for the benefit of my students, electrical and computer engineers, computer scientists, and all those who are interested in learning about Hardware Datapath Verification using Symbolic Computer Algebra and Algebraic Geometry.

I started investigating datapath verification when I was a PhD student. While at the end of my PhD work, I began to realize that commutative algebra over finite integer rings or finite fields might be an appropriate model for finite-precision arithmetic datapath verification. I reasoned: "since a k-bit vector represents integer values $\pmod{2^k}$, we should be able to model datapaths as polynomial functions over finite rings of the type $f : \mathbb{Z}_{2^k} \to \mathbb{Z}_{2^k}$ or over Galois fields $f : \mathbb{F}_{2^k} \to \mathbb{F}_{2^k}$. Then we should be able to employ some theoretical concepts from ring algebra, along with computer algebra algorithms, and derive decision procedures for verification. That would not only enable verification at the word-level, but also allow to model the bit-precise semantics of bit-vector arithmetic."

Algebra and geometry provide a unified framework to reason about systems at higher-levels of abstraction, moving from bits to k-bit words. However, to devise bit-precise and word-level verification decision procedures, there are both theoretical and practical challenges to overcome. On the one hand, the modeling of verification problems over rings $\mathbb{Z}_{2^k}$ requires commutative algebra knowledge beyond what is available in basic algebra textbooks. These rings are non-unique factorization domains, they have zero divisors, which renders most of the Euclidean-type algorithms inapplicable. On the other hand, modeling over $\mathbb{F}_{2^k}$ allows the application of algebraic geometry based concepts (*e.g.* Nullstellensatz); however, overcoming the complexity of computer algebra algorithms becomes a major challenge. In this textbook, I have tried to address both the theoretical (mathematical) and practical (computational) issues.

Unfortunately, for someone who wants to study (or explore) this area, there is no one single reference that explains the mathematical concepts, as well as *the application of these concepts* to design verification. Contemporary electrical and computer engineering education does not cover any of these concepts in its curriculum. I do not want my students and other beginners to suffer as much as I did – to search for and read papers and books, only to realize that they are not really relevant to the problem. That is one of the reasons I decided to write this book.

There is scepticism about being able to apply computer algebra to practical datapath verification problems. The scepticism is justified to some extent, in that verification of many control-dominated applications can be solved by SAT and SMT solvers, whereas computer algebra algorithms do exhibit their worst-case complexity. However, for datapath-dominated applications, our contributions have shown how these techniques can be engineered in an efficient way – by mining more information from the circuits – to solve many large word-level verification problems where SAT and SMT solvers fail. Therefore, the other reason to write this book is to educate the community about *how to apply computer algebra techniques* to solve practical verification problems.

I have attempted to write the book in a way that it is relevant both as a textbook as well as a reference for verification engineers. The book does not assume any knowledge of number theory, commutative and computer algebra, nor any knowledge of algebraic geometry. I have also tried my best to emphasize how these techniques should be applied for hardware verification. Examples are used extensively to demonstrate the concepts. I have tried not to waste time and energy on proofs of many fundamental results that are available in standard algebra textbooks. Wherever appropriate, I have tried to provide outlines of proofs and their demonstration with examples that help devise algorithms to solve verification problems.

This book is typeset using LaTeX, of course. However, the formatting of the book is borrowed entirely from Jim Hefferon's *Linear Algebra* textbook, which he has made available for free on the Internet at http://joshua.smcvt.edu/linearalgebra. Jim did an outstanding job with the page styles, margins, the formatting of headings, examples, definitions, theorems, etc., including the excellent choice of fonts. I was so impressed by his effort that I just borrowed his LaTeXstyle files to format this book. The formatting makes this mathematically intense text so much more pleasing on the eyes, and a pleasure to read. *Thank you for your efforts, Jim.*

For now, this book is being made available for free. You are free to download and use this book any which way you want! If you find any errors or typos, feel free to email them to me.

I can dedicate this book to any number of people. However, I will fill this out later.

I hope you enjoy reading the book! If you like the book, and want to boost my ego: send me a note, and forward this book to anyone else who will read it.

Some filler for the moment: Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Priyank Kalla
Electrical and Computer Engineering
University of Utah
Salt Lake City, Utah USA 84112
http://www.ece.utah.edu/~kalla

Contents

Chapter One

Galois Fields and Hardware Design

A field is a set of elements over which addition, multiplication and division by non-zero elements can be performed. Examples of fields include $\mathbb{R}, \mathbb{Q}, \mathbb{C}$, all of which have an infinite number of elements. It is also possible to construct fields with a finite number of elements. Such *finite fields* are also called *Galois fields* (GFs), and they have applications in many areas in electrical and computer engineering – such as in cryptography, error control coding, VLSI testing, etc. In fact, every digital circuit or a word-level RTL description represents a function over GFs, even though we rarely think about arbitrary digital designs in that fashion. In this chapter, we will study some fundamental properties of GFs that will educate us about hardware design and verification over GFs using algebraic geometry. Since GFs are fields, Buchberger's algorithm can be applied to compute Gröbner bases of polynomial ideals over GFs. GFs also have very interesting properties that allow the application of algebraic geometry (*e.g.* Nullstellensatz) in a mathematically elegant fashion that also makes it practical. For a more detailed study of GFs, the reader may refer to textbooks [1] and [2].

I Introduction

Galois fields are denoted as $\mathbb{F}_q$, where q corresponds to the number of elements of the field (including the elements 0, 1). The number of elements q is always

Table One.1: Additive and multiplicative inverses in $\mathbb{Z}_5$.

element	additive inverse	multiplicative inverse
0	0	undefined
1	4	1
2	3	3
3	2	2
4	1	4

a power of a prime integer, i.e. $q = p^k$, where p is a prime integer and $k \in \mathbb{Z}_{\geqslant 1}$ is an integer. Note that when $k = 1$, we have $q = p$. Then, the fields of prime cardinality p are nothing but the rings of integers $\pmod p$: i.e. $\mathbb{F}_p = \mathbb{Z}_p = \{0, 1, \ldots, p-1\}$, where addition and multiplication are performed $\pmod p$.

0.1 Example Consider the field $\mathbb{F}_5 = \mathbb{Z}_5$. The additive and multiplicative inverses of each element in $\mathbb{Z}_5$ (except 0) are also elements in $\mathbb{Z}_5$, as shown in Table One.1. In contrast, $\mathbb{Z}_4$ is not a field, as 2 does not have a multiplicative inverse in $\mathbb{Z}_4$. However, there does exist a field of 4 elements, denoted $\mathbb{F}_4$, but $\mathbb{F}_4 \neq \mathbb{Z}_4$.

We will consider the more general case of GFs of the type $\mathbb{F}_{p^k}$. Such fields are called extension fields as they are k-dimensional extensions of the base field $\mathbb{F}_p$. Of particular interest in hardware design are fields of the type $\mathbb{F}_{2^k}$ where $p = 2$ and $k > 1$. These GFs are fields of 2^k elements, and they are of interest to us because a bit-vector of size k represents 2^k distinct elements. Such fields $\mathbb{F}_{2^k}$ are also called *binary Galois extension fields* and are particularly useful in hardware design.

0.2 Definition The *characteristic* of a finite field $\mathbb{F}_q$ with unity element 1 is the smallest integer n such that $1 + \cdots + 1$ (n times) $= 0$.

The characteristic of $\mathbb{F}_p$ is, not surprisingly, the prime integer p. What is interesting is the fact that all fields $\mathbb{F}_{p^k}$ also have the characteristic p. The reason for this will become apparent in the following sections.

So how are $\mathbb{F}_{p^k}$ described? How are they constructed? What are their constituent elements? To answer these questions, let us first understand how does one construct **any** field. For this purpose, let us review integral and Euclidean domains.

II Integral and Euclidean Domains

An integral domain is a generalization of integers where no product of non-zero elements is equal to 0. More formally:

0.3 Definition An *integral domain* R is a set with two operations $(+, \cdot)$ such that:

(1) The elements of R form an abelian group under $+$ with additive identity 0.

(2) The multiplication is associative and commutative, with multiplicative identity 1.

(3) The distributive law holds: $a(b+c) = ab + ac$.

(4) The cancellation law holds: if $ab = ac$ and $a \neq 0$, then $b = c$.

Notice that the first three conditions make R a commutative ring with unity. The last condition is what makes a ring also an integral domain. Now, a Euclidean domain is an integral domain with an added feature – a notion of *size* or *degree* or a *valuation function* associated with its elements.

0.4 Definition A *Euclidean domain* $\mathbb{D}$ is an integral domain where:

(1) associated with each non-zero element $a \in \mathbb{D}$ is a non-negative integer $f(a)$ s.t. $f(a) \leqslant f(ab)$ if $b \neq 0$; and

(2) $\forall a, b\ (b \neq 0), \exists (q, r)$ s.t. $a = qb + r$, where either $r = 0$ or $f(r) < f(b)$.

Over Euclidean domains, one can apply the Euclid's algorithm to compute the GCD of a finite set of elements. Moreover, if $g = \mathrm{GCD}(g_1, \ldots, g_t)$ then g can be written as a linear combination of $g_1, \ldots, g_t$ as $g = \sum_i u_i g_i$.

0.5 Remark While studying Gröbner bases over univariate polynomial rings with coefficients from a field (which are Euclidean domains!), we have seen that the Gröbner basis of a set of polynomials $\{g_1, \ldots, g_t\}$ is their GCD: $g = gcd(g_1, \ldots, g_t)$. Obviously, g can be represented as a combination of $g_1, \ldots, g_t$ as $g \in J = \langle g_1, \ldots, g_t \rangle$.

If two elements $a, b \in \mathbb{D}$ are relatively prime, then their $gcd(a, b) = 1$. Every non-unit element of a Euclidean domain can be uniquely factorized as a product of (powers of) primes. This should not be surprising as Euclidean domains are also unique factorization domains.

Some examples that differentiate integral domains from Euclidean domains:

0.6 Example Let $\mathbb{D} = \mathbb{Z}$, and let $g_1 = 39, g_2 = 65$, their gcd is 13, and $13 = 2 \cdot 39 + (-1) \cdot 65$, a linear combination of g_1, g_2. The set $\mathbb{Z}$ is both an integral and a Euclidean domain.

0.7 Example The ring $\mathbb{F}[x]$ is a Euclidean domain where $\mathbb{F}$ is any field. The *size* of all non-zero elements (polynomials) in $\mathbb{F}[x]$ is the degree of the polynomial.

0.8 Example Now consider the polynomial ring $\mathbb{R}[x, y]$. It is an integral domain, but it is **not** a Euclidean domain. As a counter-example, consider elements $x, y \in \mathbb{R}[x, y]$. They are relatively prime, so their $gcd(x, y) = 1$. It is not possible to write 1 as a linear combination of x and y: i.e. $f_1(x, y) \cdot x + f_2(x, y) \cdot y = 1$ is not possible if $f_1, f_2 \in \mathbb{R}[x, y]$.

0.9 Example The finite integer rings $\mathbb{Z}_{2^k}, k > 1$ are neither integral domains nor Euclidean domains, as they contain zero divisors. As an example, consider the ring $\mathbb{Z}_8$, where elements $4 \neq 0, 2 \neq 0$ but $4 \cdot 2 = 0$.

With this knowledge, we are now ready to understand extension fields.

III Constructing Extension Fields

0.10 Definition Let $\mathbb{D}$ be a Euclidean domain, and $p \in \mathbb{D}$ be a prime element. Then $\mathbb{D} \pmod{p}$ is a field.

So, when $\mathbb{D} = \mathbb{Z}$, $\mathbb{Z} \pmod{p} = \mathbb{Z}_p$ is a field (and Def. 0.10 is precisely the reason why $\mathbb{Z}_p$ is field!). The univariate polynomial ring $\mathbb{R}[x]$ is also a Euclidean domain. So, if we take a prime element p from $\mathbb{R}[x]$ and compute $\mathbb{R}[x] \pmod{p}$, we will obtain a field. The meaning of "$\mathbb{R}[x] \pmod{p}$" is: take any and all elements (polynomials) from $\mathbb{R}[x]$, divide by p, and take the remainders. In other words, $\mathbb{R}[x] \pmod{p} = \{f(x) \mid \forall g(x) \in \mathbb{R}[x], f(x) = g(x) \pmod{p}\}$.

0.11 Remark In the case of polynomials, the term *prime* is usually replaced with *irreducible*. A prime element in a polynomial ring is nothing but an *irreducible polynomial*, which cannot be factored into a product of two or more polynomials of positive degree.

Notice that x^2+1 is irreducible over $\mathbb{R}$. So, let us compute $\mathbb{R}[x] \pmod{x^2+1}$. This generates nothing but the field of complex numbers: $\mathbb{C} = \mathbb{R}[x] \pmod{x^2+1}$.

Not convinced? Then, consider any element $f(x) \in \mathbb{R}[x] \pmod{x^2+1}$. Since f is the remainder of division by x^2+1, it has to be a linear expression of the form $f = ax + b$ where the coefficients $a, b \in \mathbb{R}$. Similarly, let $g = cx + d \in \mathbb{R}[x]$ $\pmod{x^2+1}$. Multiply f, g keeping in mind that we are operating in $\mathbb{R}[x]$ $\pmod{x^2+1}$, where every result has to be reduced (divided by) x^2+1. Then:

$$\begin{aligned} f \cdot g &= (ax+b)(cx+d) \pmod{x^2+1} \\ &= acx^2 + (ad+bc)x + bd \pmod{x^2+1} \\ &= (ad+bc)x + (bd-ac) \text{ after reducing by } x^2 = -1 \end{aligned}$$

Replace x with i, and you will notice that we just multiplied $(ai+b)(ci+d)$ with $i^2 = -1$. So, in conclusion $\mathbb{C} = \mathbb{R}[x] \pmod{x^2+1}$, where $\mathbb{C}$ is a 2-dimensional extension of the base field $\mathbb{R}$, where $2 = \text{degree}(x^2+1)$.

Now the following set inclusions should become amply clear: Rings $\supset$ Integral Domains $\supset$ Unique Factorization Domains $\supset$ Euclidean Domains $\supset$ Fields

III.1 Galois Extension Fields

Galois fields $\mathbb{F}_{p^k}$ are also k-dimensional extensions of the base field $\mathbb{F}_p = \mathbb{Z}_p$. Consider the ring $\mathbb{F}_p[x]$ and let $P(x) \in \mathbb{F}_p[x]$ be an irreducible polynomial of degree k. Then $\mathbb{F}_{p^k} = \mathbb{F}_p[x] \pmod{P(x)}$.

Let us focus on $\mathbb{F}_{2^k} = \mathbb{F}_2[x] \pmod{P(x)}$, where $P(x)$ is the irreducible polynomial of degree k. Note that $P(x) \in \mathbb{F}_2[x]$, so its coefficients are in $\{0, 1\}$. Irreducible polynomials of any degree k always exist, so $\mathbb{F}_{2^k}$ can be constructed for arbitrary $k \geqslant 1$. Given in the table below are some examples of irreducible polynomials in $\mathbb{F}_2[x]$ for degree $k = 1, \ldots, 4$.

Table One.2: Some irreducible polynomials in $\mathbb{F}_2[x]$.

Degree	Irreducible Polynomials
1	$x; x+1$
2	x^2+x+1
3	$x^3+x+1; x^3+x^2+1$
4	$x^4+x+1; x^4+x^3+1; x^4+x^3+x^2+x+1$

Coming back to $\mathbb{F}_{2^k} = \mathbb{F}_2[x] \pmod{P(x)}$, let α be a root of $P(x)$, i.e. $P(\alpha) = 0$.

Note that $P(x)$ has no roots in $\mathbb{F}_2$ as it is irreducible in $\mathbb{F}_2[x]$; however, the root lies in its algebraic extension $\mathbb{F}_{2^k}$. Any element $A \in \mathbb{F}_{2^k}$ can therefore be represented as:

$$A = \sum_{i=0}^{k-1} (a_i \cdot \alpha^i) = a_0 + a_1 \cdot \alpha + \cdots + a_{k-1} \cdot \alpha^{k-1} \qquad \text{(One.1)}$$

where $a_i \in \mathbb{F}_2$ are the coefficients and $P(\alpha) = 0$. The degree of any element A in $\mathbb{F}_{2^k}$ is always less than k. This is because A is always computed modulo $P(x)$, and $P(x)$ has degree k. The remainder $\pmod{P(x)}$ can be of degree at most $k-1$. For this reason, the field $\mathbb{F}_{2^k}$ can be viewed as a k-dimensional vector space over $\mathbb{F}_2$. The example below explains the construction of $\mathbb{F}_{2^4}$.

1.1 Example Let us construct $\mathbb{F}_{2^4}$ as $\mathbb{F}_2[x] \pmod{P(x)}$, where $P(x) = x^4+x^3+1 \in \mathbb{F}_2[x]$ is an irreducible polynomial of degree $k = 4$. Let α be a root of $P(x)$, i.e. $P(\alpha) = 0$.

Any element $A \in \mathbb{F}_2[x] \pmod{x^4 + x^3 + 1}$ has a representation of the type: $A = a_3x^3 + a_2x^2 + a_1x + a_0$ (degree < 4) where the coefficients $a_3, \dots, a_0$ are in $F_2 = \{0, 1\}$. Since there are only 16 such polynomials, we obtain 16 elements in the field $\mathbb{F}_{2^4}$. Each element in $\mathbb{F}_{2^4}$ can then be viewed as a 4-bit vector over $\mathbb{F}_2$: $\mathbb{F}_{2^4}$={(0000), (0001), ... (1110),(1111)}. If α is a root of $P(x)$, then each element also has an exponential representation; all three representations are shown in Table One.3. For example, consider the element α^{12}. Computing $\alpha^{12} \pmod{\alpha^4 + \alpha^3 + 1} = \alpha + 1 = (0011)$; hence we have the three equivalent representations.

Table One.3: Bit-vector, Exponential and Polynomial representation of elements in $\mathbb{F}_{2^4} = \mathbb{F}_2[x] \pmod{x^4 + x^3 + 1}$

$a_3a_2a_1a_0$	Exponential	Polynomial	$a_3a_2a_1a_0$	Exponential	Polynomial
0000	0	0	1000	α^3	α^3
0001	1	1	1001	α^4	α^3+1
0010	α	α	1010	α^{10}	$\alpha^3+\alpha$
0011	α^{12}	$\alpha+1$	1011	α^5	$\alpha^3+\alpha+1$
0100	α^2	α^2	1100	α^{14}	$\alpha^3+\alpha^2$
0101	α^9	α^2+1	1101	α^{11}	$\alpha^3+\alpha^2+1$
0110	α^{13}	$\alpha^2+\alpha$	1110	α^8	$\alpha^3+\alpha^2+\alpha$
0111	α^7	$\alpha^2+\alpha+1$	1111	α^6	$\alpha^3+\alpha^2+\alpha+1$

1.2 Remark It should be clear by now that the characteristic of $\mathbb{F}_{p^k}$ is the prime integer p. This is because $\mathbb{F}_{p^k} \equiv \mathbb{F}_p[x] \pmod{P(x)}$; since all coefficients in $\mathbb{F}_p[x] \pmod{P(x)}$ are in $\mathbb{F}_p$. Therefore, for binary GFs $\mathbb{F}_{2^k} = \mathbb{F}_2[x] \pmod{P(x)}$, all coefficients are reduced modulo 2. Consequently, $-1 = +1$ in all fields $\mathbb{F}_{2^k}$.

1.3 Example (Addition and Multiplication over $\mathbb{F}_{2^k}$) From Example 1.1, let us demonstrate addition and multiplication of GF elements. Addition of $\alpha^5 + \alpha^{11}$

$$\begin{aligned}
\alpha^5 + \alpha^{11} &= \alpha^3 + \alpha + 1 + \alpha^3 + \alpha^2 + 1 \\
&= 2 \cdot \alpha^3 + \alpha^2 + \alpha + 2 \\
&= \alpha^2 + \alpha \quad \text{(as characteristic of } \mathbb{F}_{2^k} = 2) \\
&= \alpha^{13}
\end{aligned}$$

Observe that this addition is nothing but a bit-vector XOR operation. Multiplication can be performed using both the exponential and polynomial representations. For example, $\alpha^4 \cdot \alpha^{10} = \alpha^{14} = \alpha^3 + \alpha^2$ when reduced modulo $\pmod{\alpha^4 + \alpha^3 + 1}$. Likewise:

$$\begin{aligned}
\alpha^4 \cdot \alpha^{10} &= (\alpha^3 + 1)(\alpha^3 + \alpha) \\
&= \alpha^6 + \alpha^4 + \alpha^3 + \alpha \\
&= \alpha^4 \cdot \alpha^2 + (\alpha^4 + \alpha^3) + \alpha \\
&= (\alpha^3 + 1) \cdot \alpha^2 + (1) + \alpha \quad (\text{as } \ \alpha^4 = \alpha^3 + 1) \\
&= \alpha^5 + \alpha^2 + \alpha + 1 \\
&= \alpha^4 \cdot \alpha + \alpha^2 + \alpha + 1 \\
&= (\alpha^3 + 1) \cdot \alpha + \alpha^2 + \alpha + 1 \\
&= \alpha^4 + \alpha^2 + 1 \\
&= \alpha^3 + \alpha^2
\end{aligned}$$

Just keep in mind that $P(\alpha) = \alpha^4 + \alpha^3 + 1 = 0$ implies that $\alpha^4 = -(\alpha^3 + 1) = (\alpha^3 + 1)$, as $-1 = +1$ in fields of characteristic 2.

The inverse of any element α can be computed using the extended Euclidean algorithm. Euclid's algorithm is used to compute the GCD of two elements a, b; the algorithm can be used to find p_1, p_2 such that $a \cdot p_1 + b \cdot p_2 = \text{GCD}(a, b)$ as the $\text{GCD}(a, b)$ can be written as a linear combination of a, b. When the field $\mathbb{F}_p = \mathbb{Z}_p$, then the inverse of an element a can be computed using the Euclidean algorithm with arguments (a, p):

$$\begin{aligned} a \cdot p_1 + p \cdot p_2 &= \textsc{gcd}(a, p) \\ a \cdot p_1 + p \cdot p_2 &= 1 \pmod{p}, \quad \text{as } a \text{ and } p \text{ are co-prime} \\ a \cdot p_1 + 0 &= 1 \pmod{p} \end{aligned}$$

So, a and p_1 are inverses of each other. Similarly, to find the inverse of any arbitrary element $\beta \in \mathbb{F}_{2^k}$, compute $\textsc{gcd}(\beta, P(\alpha))$ using the Euclidean algorithm to find:

$$\begin{aligned} \beta \cdot p_1 + P(\alpha) \cdot p_2 &= \textsc{gcd}(\beta, P(\alpha)) \\ \beta \cdot p_1 + P(\alpha) \cdot p_2 &= 1 \ \ (\text{as } P(x) \text{ is irreducible}) \\ \beta \cdot p_1 &= 1 \end{aligned}$$

where β and p_1 are inverses of each other.

There may exist more than one irreducible polynomials with degree k, as observed in Table One.2. In such cases, any degree k irreducible polynomial can be used for field construction. For example, both $x^3 + x^2 + 1$ and $x^3 + x + 1$ are irreducible in $\mathbb{F}_2$ and either one can be used to construct $\mathbb{F}_{2^3}$. This is due to the following result:

1.4 Theorem There exists a **unique** field $\mathbb{F}_{p^k}$, for any prime p and any positive integer k.

Theorem 1.4 implies that finite fields with the same number of elements are isomorphic to each other *up to the labeling of the elements.* So, different irreducible polynomials only lead to different labeling of the elements. This concept will become very clear when we study conjugates and minimal polynomials.

III.2 Vanishing Polynomials of $\mathbb{F}_q$

2.1 Lemma Let A be any non-zero element in $\mathbb{F}_q$, then $A^{q-1} = 1$.

As a consequence of Lemma 2.1, the following is a very important result that we will use to investigate solutions to polynomial equations in $\mathbb{F}_q$.

2.2 Theorem [Generalized Fermat's Little Theorem] Given a finite field $\mathbb{F}_q$, each element $A \in \mathbb{F}_q$ satisfies:

$$\begin{aligned} A^q &\equiv A \\ A^q - A &\equiv 0 \end{aligned} \tag{One.2}$$

As a polynomial extension of the above consequence, let $x^q - x$ be a polynomial in $\mathbb{F}_q[x]$. Every element $A \in \mathbb{F}_q$ is a solution to $x^q - x = 0$. Therefore, $x^q - x$ always *vanishes* in $\mathbb{F}_q$, and such polynomials are called **vanishing polynomials** of the field $\mathbb{F}_q$.

2.3 Example Given $\mathbb{F}_{2^2} = \{0, 1, \alpha, \alpha + 1\}$ with $P(x) = x^2 + x + 1$, where $P(\alpha) = 0$.

$$\begin{aligned} 0^{2^2} &= 0 \\ 1^{2^2} &= 1 \\ \alpha^{2^2} &= \alpha \pmod{\alpha^2 + \alpha + 1} \\ (\alpha + 1)^{2^2} &= \alpha + 1 \pmod{\alpha^2 + \alpha + 1} \end{aligned}$$

Every element of $\mathbb{F}_q$ is a solution to vanishing polynomials $x^q - x = 0$. Therefore, the variety $V_{\mathbb{F}_q}(x^q - x) = \mathbb{F}_q$ itself. Note that the variety is being considered over $\mathbb{F}_q$ and not over the algebraic closure $\overline{\mathbb{F}_q}$. By the way, we will denote the ideal generated by the vanishing polynomial as $J_0 = \langle x^q - x \rangle$, so that $V_{\mathbb{F}_q}(J_0) = \mathbb{F}_q$. Similarly, over multivariate polynomial rings, let $J_0 = \langle x_1^q - x_1, x_2^q - x_2, \ldots, x_n^q - x_n \rangle \subset \mathbb{F}_q[x_1, \ldots, x_n]$; then $V_{\mathbb{F}_q}(J_0) = (\mathbb{F}_q)^n$.

IV Irreducible and Primitive Polynomials

As shown in Table One.2, there may be more than one irreducible polynomials in $\mathbb{F}_2[x]$. Some irreducible polynomials have an important property in that they can generate all non-zero elements of the field. Such irreducible polynomials are called primitive polynomials. If α is a root of the primitive polynomial, then $\mathbb{F}_q = \{0, 1 = \alpha^{q-1}, \alpha, \alpha^2, \alpha^3, \ldots, \alpha^{q-2}\}$. Here α is called the primitive element. This property also highlights the multiplicative group structure of the field: that all the non-zero elements of the field form a group under multiplication. Actually, this group is also cyclic, in that multiplication by α generates all the elements and repeats. Both primitive and non-primitive polynomials can be used for field construction, but if you have a choice, using a primitive polynomial might be beneficial in this regard. A primitive polynomial $P(x)$ of degree k has the property that the smallest integer n for which $P(x) \mid (x^n + 1)$ is $n = 2^k - 1$.

0.4 Example In Example 1.1, the irreducible polynomial $x^4 + x^3 + 1$ is primitive; which is why every element of $\mathbb{F}_{2^4}$ is a power of α (see the exponential representation), and every non-zero element of the field can be generated using powers of α. On the other hand, the irreducible polynomial $P_1(x) = x^4 + x^3 + x^2 + x + 1$ is not primitive because if $P_1(\alpha) = 0$ then:

$$\begin{aligned}
\alpha^4 &= \alpha^3 + \alpha^2 + \alpha + 1 \\
\alpha^5 &= \alpha^4 \cdot \alpha \\
&= (\alpha^3 + \alpha^2 + \alpha + 1)(\alpha) \\
&= (\alpha^4) + \alpha^3 + \alpha^2 + \alpha \\
&= (\alpha^3 + \alpha^2 + \alpha + 1) + (\alpha^3 + \alpha^2 + \alpha) \\
&= 1
\end{aligned}$$

So, powers of α can only generate a few elements, not all.

IV.1 Conjugates and Minimal Polynomials

Let $f(x) \in \mathbb{F}_2[x]$ be an arbitrary polynomial, and let β be an element in $\mathbb{F}_{2^k}$ for any $k > 1$. If β is a root of $f(x)$, then for any $l \geqslant 0, \beta^{2^l}$ is also a root of $f(x)$. The elements β^{2^l} are called conjugates of β. Since the field is finite, the number of distinct conjugates is also finite. Raising β to powers 2^l for $l = 0, 1, 2, \dots$ will cause the conjugates to repeat.

1.1 Example Let $\mathbb{F}_{16} = \mathbb{F}_2[x] \pmod{P(x) = x^4 + x^3 + 1}$. Let $P(\alpha) = 0$. Let us find conjugates of α as α^{2^l}.

$$\begin{aligned}
&l = 1 : \alpha^2 \\
&l = 2 : \alpha^4 = \alpha^3 + 1 \\
&l = 3 : \alpha^8 = \alpha^3 + \alpha^2 + \alpha \\
&l = 4 : \alpha^{16} = \alpha \;\; \text{(conjugates start to repeat)}
\end{aligned}$$

So $\alpha, \alpha^2, \alpha^3+1, \alpha^3+\alpha^2+\alpha$ are conjugates of each other.

Let e be the smallest integer such that $\beta^{2^e} = \beta$. Construct the polynomial $f(x) = \prod_{i=0}^{e-1}(x+\beta^{2^i})$. Then $f(x)$ is an irreducible polynomial, and it is also called the **minimal polynomial** of β.

1.2 Example Following on from Example 1.1, let us construct the minimal polynomial of α, where $P(\alpha) = \alpha^4+\alpha^3+1 = 0$. We obtain $f(x) = (x+\alpha)(x+\alpha^2)(x+\alpha^3+1)(x+\alpha^3+\alpha^2+\alpha)$. Simplifying $f(x) \pmod{P(x)}$, we obtain $f(x) = x^4+x^3+1$. Is it surprising that you obtained the irreducible polynomial that was used for field construction? After all, α is a root of this polynomial by design, and all the conjugates of α are also its roots.

Now, let us take the element $\beta = \alpha^3$. Its conjugates are α^6, α^{12} and α^{24}. Construct $f(x) = (x+\alpha^3)(x+\alpha^6)(x+\alpha^{12})(x+\alpha^{24})$, simplify $\pmod{P(\alpha) = \alpha^4+\alpha^3+1}$, and you will observe that $f(x) = x^4+x^3+x^2+x+1$. Refer to table One.2 and observe that this is the other irreducible polynomial of degree 4.

Continuing in this fashion, consider element α^7. Its conjugates are $\alpha^{14}, \alpha^{28}, \alpha^{56}$. The minimal polynomial of these elements is $f(x) = (x+\alpha^7)(x+\alpha^{14})(x+\alpha^{28})(x+\alpha^{56}) = x^4+x+1$. From Table One.2, this is the third irreducible polynomial.

Finally, let us now address the more interesting case – that of the element α^5 and its conjugates, $(\alpha^5)^{2^l}$ for $l = 0, 1, \ldots$:

$$\begin{aligned}
l = 0 :& (\alpha^5)^{2^l} = \alpha^5\\
l = 1 :& (\alpha^5)^{2^l} = \alpha^{10}\\
l = 2 :& (\alpha^5)^{2^l} = \alpha^{20}\\
&= \alpha^{15}\cdot\alpha^5\\
&= (1)\cdot\alpha^5 \quad \text{(due to Lemma 2.1)}
\end{aligned}$$

This means α^5 has only one other conjugate α^{10}. Their minimal polynomial $f(x) = (x+\alpha^5)(x+\alpha^{10}) = x^2+x+1$. This is not a degree-4 polynomial, rather a degree-2 polynomial. In fact, it is the irreducible used to construct $\mathbb{F}_4 = \mathbb{F}_2[x] \pmod{x^2+x+1}$. Similarly, $f(x) = x+1$ is the minimal polynomial of unity element (1). This implies that the elements α^5 and α^{10} are also elements of $\mathbb{F}_4$ such that $\mathbb{F}_2 \subset \mathbb{F}_4 \subset \mathbb{F}_{16}$. This is shown in Fig. One.1.

In this fashion, all the irreducible polynomials associated with all the elements of $\mathbb{F}_{16}$ can be generated, which can also give you the information about all the fields that are contained in it.

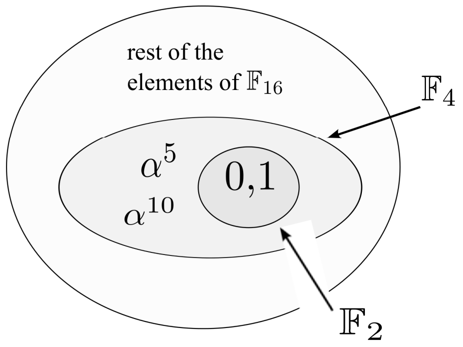


Figure One.1: Containment of fields: $\mathbb{F}_2 \subset \mathbb{F}_4 \subset \mathbb{F}_{16}$

IV.2 Containment and Algebraic Closure of $\mathbb{F}_{2^k}$

Fig. One.1 shows that $\mathbb{F}_2 \subset \mathbb{F}_4 \subset \mathbb{F}_{16}$, which also shows the containment of elements. A field is closed under addition and multiplication; this fact can also be demonstrated on these elements. Addition $\alpha^5 + \alpha^{10} = 1$ and multiplication $\alpha^5 \cdot \alpha^{10} = 1$ also leads to elements within $\mathbb{F}_4$. Containment of Galois fields is based on the following result:

2.1 Theorem $\mathbb{F}_{2^n} \subset \mathbb{F}_{2^m}$ if n divides m.

Therefore:

- $\mathbb{F}_2 \subset \mathbb{F}_{2^2} \subset \mathbb{F}_{2^4} \subset \mathbb{F}_{2^8} \dots,$
- $\mathbb{F}_2 \subset \mathbb{F}_{2^3} \subset \mathbb{F}_{2^6} \subset \dots,$
- $\mathbb{F}_2 \subset \mathbb{F}_{2^5} \subset \mathbb{F}_{2^{10}} \subset \dots,$ and so on.

The algebraic closure of the Galois field $\mathbb{F}_{2^k}$ is the union of all fields $\mathbb{F}_{2^n}$ such that $k \mid n$.

Let us revisit the issue of vanishing polynomials $x^q - x$, and look at where do their solutions lie. Let $\mathbb{F}_q$ denote the field, $\overline{\mathbb{F}_q}$ denote its algebraic closure, and denote by $J_0 = \langle x^q - x \rangle$ the ideal of vanishing polynomials. It is clear that the variety $V_{\mathbb{F}_q}(J_0) = \mathbb{F}_q$. What is also interesting is that the variety of vanishing polynomials does not change over the field or over the closure:

2.2 Lemma

$$V_{\overline{\mathbb{F}_q}}(J_0) = V_{\mathbb{F}_q}(J_0) = V(x^q - x) = \mathbb{F}_q \qquad \text{(One.3)}$$

2.3 Example Consider $\alpha^5 \in \mathbb{F}_{16}$. Due to Theorem 2.2, $(\alpha^5)^{16} = \alpha^5$. However, we also saw in the previous example that $\alpha^5 \in \mathbb{F}_4$. Noting that $\mathbb{F}_{16}$ is in the algebraic closure of $\mathbb{F}_4$ as $\mathbb{F}_4 \subset \mathbb{F}_{16}$, we also observe that $(\alpha^5)^4 = \alpha^5$.

With this information, we are now ready to study hardware designs over Galois fields.

V Hardware Implementations of Arithmetic over $\mathbb{F}_{2^k}$

In some cases, finite field (primitive) computations such as ADD, MUL, etc., are implemented in hardware, and algorithms are then implemented in software (e.g. cryptoprocessors [3] [4]). In other cases, the entire design can be implemented in hardware – such as a one-shot Reed-Solomon encoder-decoder chip [5] [6], or the point multiplication circuitry [7] used in elliptic curve cryptosystems. Therefore, there has been a lot of research in VLSI implementations of finite field arithmetic. We describe the design of such primitive computations to shed some light on the architectures and their design and verification complexity.

Addition in $\mathbb{F}_{2^k}$ is performed by correspondingly adding the polynomials together, and reducing the coefficients of the result modulo the characteristic 2.

0.4 Example Given $A = \alpha^3 + \alpha^2 + 1 = (1101)$ and $B = \alpha^2 + 1 = (0101)$ in $\mathbb{F}_{2^4}$,

$$\begin{aligned} A + B &= (\alpha^3 + \alpha^2 + 1) + (\alpha^2 + 1) \\ &= (\alpha^3) + (\alpha^2 + \alpha^2) + (1 + 1) \\ &= \alpha^3 = (1000) \end{aligned}$$

0.5 Example A 4-bit adder in $\mathbb{F}_{2^4}$ is given in Figure One.2. It takes as inputs two 4-bit vectors: $A = (a_3 a_2 a_1 a_0)$, $B = (b_3 b_2 b_1 b_0)$ and computes the result $Z = (z_3 z_2 z_1 z_0)$. Note an adder circuit is trivial and only consists of *XOR* gates.

Conceptually, the multiplication $Z = A \times B \pmod{P(x)}$ in $\mathbb{F}_{2^k}$ consists of two steps. In the first step, the multiplication $A \times B$ is performed, and in the second step, the result is reduced modulo the irreducible polynomial $P(x)$. Multiplication procedure is shown in Example 0.6.

0.6 Example Consider the field $\mathbb{F}_{2^4}$. We take as inputs: $A = a_0 + a_1 \cdot \alpha +$

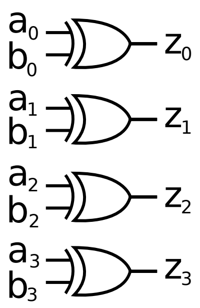


Figure One.2: 4-bit adder over $\mathbb{F}_{2^4}$.

$a_2 \cdot \alpha^2 + a_3 \cdot \alpha^3$ and $B = b_0 + b_1 \cdot \alpha + b_2 \cdot \alpha^2 + b_3 \cdot \alpha^3$, along with the irreducible polynomial $P(x) = x^4 + x^3 + 1$. We have to perform the multiplication $Z = A \times B \pmod{P(x)}$. The coefficients of $A = \{a_0, \dots, a_3\}, B = \{b_0, \dots, b_3\}$ are in $\mathbb{F}_2 = \{0, 1\}$. Multiplication can be performed as shown below:

$$
\begin{array}{ccccccc}
 & & & a_3 & a_2 & a_1 & a_0 \\
\times & & & b_3 & b_2 & b_1 & b_0 \\
\hline
 & & & a_3 \cdot b_0 & a_2 \cdot b_0 & a_1 \cdot b_0 & a_0 \cdot b_0 \\
 & & a_3 \cdot b_1 & a_2 \cdot b_1 & a_1 \cdot b_1 & a_0 \cdot b_1 & \\
 & a_3 \cdot b_2 & a_2 \cdot b_2 & a_1 \cdot b_2 & a_0 \cdot b_2 & & \\
a_3 \cdot b_3 & a_2 \cdot b_3 & a_1 \cdot b_3 & a_0 \cdot b_3 & & & \\
\hline
s_6 & s_5 & s_4 & s_3 & s_2 & s_1 & s_0
\end{array}
$$

The result $Sum = s_0 + s_1 \cdot \alpha + s_2 \cdot \alpha^2 + s_3 \cdot \alpha^3 + s_4 \cdot \alpha^4 + s_5 \cdot \alpha^5 + s_6 \cdot \alpha^6$,

$$
\begin{aligned}
s_0 &= a_0 \cdot b_0 \\
s_1 &= a_0 \cdot b_1 + a_1 \cdot b_0 \\
s_2 &= a_0 \cdot b_2 + a_1 \cdot b_1 + a_2 \cdot b_0 \\
s_3 &= a_0 \cdot b_3 + a_1 \cdot b_2 + a_2 \cdot b_2 + a_3 \cdot b_1 \\
s_4 &= a_1 \cdot b_3 + a_2 \cdot b_1 + a_3 \cdot b_1 \\
s_5 &= a_2 \cdot b_3 + a_3 \cdot b_2 \\
s_6 &= a_3 \cdot b_3
\end{aligned}
$$

Here the multiply "$\cdot$" and add "$+$" operations are performed modulo 2, so they can be implemented in a circuit using AND and XOR gates. Note that unlike integer multipliers, there are no carry-chains in the design, as the coefficients are always reduced modulo $p = 2$. However, the result is yet to be reduced modulo the primitive polynomial $P(x) = x^4 + x^3 + 1$. This is shown below, where higher degree coefficients are reduced $\pmod{P(x)}$.

s_3	s_2	s_1	s_0	
s_4	0	0	s_4	$s_4 \cdot \alpha^4 \pmod{P(\alpha)} = s_4 \cdot (\alpha^3 + 1)$
s_5	0	s_5	s_5	$s_5 \cdot \alpha^5 \pmod{P(\alpha)} = s_5 \cdot (\alpha^3 + \alpha + 1)$
s_6	s_6	s_6	s_6	$s_6 \cdot \alpha^6 \pmod{P(\alpha)} = s_6 \cdot (\alpha^3 + \alpha^2 + \alpha + 1)$
z_3	z_2	z_1	z_0	

The final result (output) of the circuit is: $Z = z_0 + z_1\alpha + z_2\alpha^2 + z_3\alpha^3$; where $z_0 = s_0 + s_4 + s_5 + s_6$; $z_1 = s_1 + s_5 + s_6$; $z_2 = s_2 + s_6$; $z_3 = s_3 + s_4 + s_5 + s_6$.

The above multiplier design is called the *Mastrovito multiplier* [8] which is the most straightforward way to design a multiplier over $\mathbb{F}_{2^k}$. A logic circuit for a 4-bit *Mastrovito* multiplier over *finite field* $\mathbb{F}_{2^4}$ is illustrated in Fig. One.3.

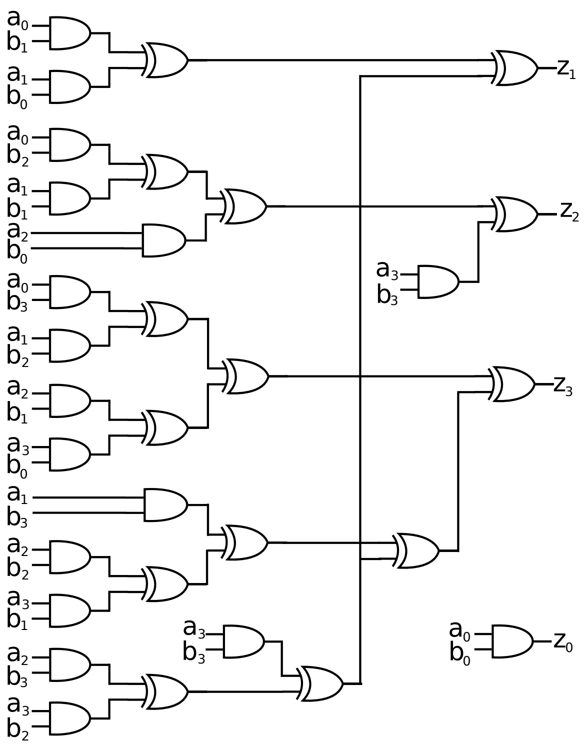


Figure One.3: Mastrovito multiplier over $\mathbb{F}_{2^4}$.

Modular multiplication is at the heart of many public-key cryptosystems, such as Elliptic Curve Cryptography (ECC) [9]. Due to the very large field size (and hence the datapath width) used in these cryptosystems, the above *Mastrovito* multiplier architecture is inefficient, especially when exponentiation and repeat multiplications are performed. Therefore, efficient hardware and

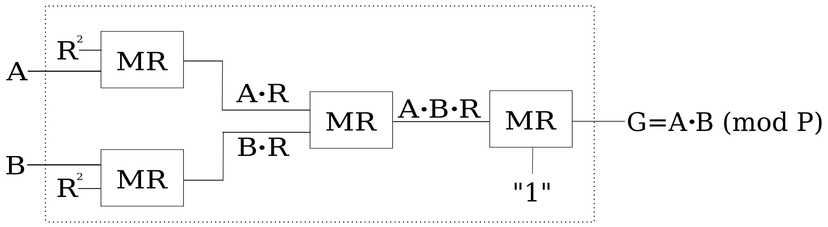


Figure One.4: *Montgomery* multiplier over $\mathbb{F}_{2^k}$

software implementations of modular multiplication algorithms are used to overcome the complexity of such operations. These include the Montgomery reduction [10] [11] and the Barrett reduction [12].

Montgomery Reduction: Montgomery reduction (MR) computes:

$$G = MR(A, B) = A \cdot B \cdot R^{-1} \pmod{P(x)} \tag{One.4}$$

where A, B are k-bit inputs, $R = \alpha^k$, R^{-1} is multiplicative inverse of R in $\mathbb{F}_{2^k}$, and $P(x)$ is the irreducible polynomial for $\mathbb{F}_{2^k}$. In general, R is a power of a base; over GFs, R is usually selected as $R = \alpha^k$. Since Montgomery reduction cannot directly compute $A \cdot B$, to compute $A \cdot B \pmod{P(x)}$, we need to pre-compute $A \cdot R$ and $B \cdot R$, as shown in Fig. One.4.

Each *MR* block in Fig. One.4 represents a Montgomery reduction step which is a hardware implementation of the algorithm shown in Algorithm 1.

Algorithm 1: Montgomery Reduction Algorithm [11]

```
Input: $A(x), B(x) \in \mathbb{F}_{2^k}$; irreducible polynomial $P(x)$.
Output: $G(x) = A(x) \cdot B(x) \cdot x^{-k} \pmod{P(x)}$.
$G(x) := 0$
for ($i = 0; i \leqslant k-1; ++i$) do
    $G(x) := G(x) + A_i \cdot B(x)$ /*$A_i$ is the $i^{th}$ bit of $A$*/;
    $G(x) := G(x) + G_0 \cdot P(x)$ /*$G_0$ is the lowest bit of G*/;
    $G(x) := G(x)/x$ /*Right shift 1 bit*/;
end
```

The design of Fig. One.4 is an overkill to compute just $A \cdot B \pmod{P(x)}$. However, when these multiplications are performed repeatedly, such as in iterative squaring, then the Montgomery approach speeds-up the computation. As shown in [13], the critical path delay and gate counts of a squarer designed using the Montgomery approach are much smaller than the traditional approaches.

Barrett Reduction: Barrett reduction is the other widely used multiplier design method adopted in cryptography system designs. Similar to Montgomery reduction, the traditional Barrett reduction, proposed in [14], needs a pre-computed value of the reciprocal/inverse of modulus $P(x)$. This pre-computation requires extra computational time and memory space. To overcome this limitation, the recent approach of [12] avoids such a pre-computation of inverses and therefore greatly simplifies the hardware design implementation. This algorithmic computation is shown in Algorithm 2.

Algorithm 2: Barrett Reduction Without Pre-Computation Algorithm [12]

```
Input: $R(x) \in \mathbb{F}_{2^k}$; irreducible polynomial $P(x) = x^n + \sum_{i=0}^{l} m_i \cdot x^i$
    satisfying $l = \lfloor \frac{n}{2} \rfloor, m_i \in \{0,1\}$.
Output: $G(x) = R(x) \pmod{P(x)}$.
$Q_1(x) = \frac{R(x)}{x^n}$ /*Right shift $n$ bit*/;
$Q_2(x) = P(x) \cdot Q_1(x)$ ;
$Q_3(x) = \frac{Q_2(x)}{x^n}$ ;
$G_1(x) = R(x) \pmod{x^n}$ /*Keep the lower $n$ bits of $R(x)$*/;
$G_2(x) = P(x) \cdot Q_3(x) \pmod{x^n}$ ;
$G(x) = G_1(x) + G_2(x)$ ;
```

Based on Barrett reduction, a multiplier can be designed with two simple steps: multiplication $R = A \times B$ and a subsequent Barrett reduction $G = R \pmod{P}$. This is shown in Fig. One.5. As we can see, a Barrett multiplier is similar to a Mastrovito multiplier except for the reduction step.

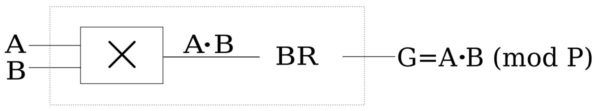


Figure One.5: Barrett multiplier over $\mathbb{F}_{2^k}$.

One of the most influential applications of finite fields is in elliptic curve cryptography (ECC). ECC is an approach to public-key cryptography based on the algebraic structure of elliptic curves over finite fields. The main operations of encryption, decryption and authentication in ECC rely on *point multiplications*. Point multiplication involves a series of addition and doubling of points on the elliptic curve. A drawback of traditional point multiplication is that each point addition and doubling involves a multiplicative inverse operation over finite fields.

Representing the points in projective coordinate systems [7] eliminates the need for multiplicative inverse operation and therefore increases the efficiency of point multiplication operation. Let us consider designs based on the López-Dahab (LD) coordinate system [15]:

0.7 Example Consider point addition in LD projective coordinate. Given an elliptic curve: $Y^2 + XYZ = X^3Z + aX^2Z^2 + bZ^4$ over $\mathbb{F}_{2^k}$, where X, Y, Z are k-bit vectors that are elements in $\mathbb{F}_{2^k}$ and similarly, a, b are constants from the field. Let $(X_1, Y_1, Z_1) + (X_2, Y_2, 1) = (X_3, Y_3, Z_3)$ represent point addition over the elliptic curve. Then X_3, Y_3, Z_3 can be computed as follows:

$$\begin{aligned}
A &= Y_2 \cdot Z_1^2 + Y_1 \\
B &= X_2 \cdot Z_1 + X_1 \\
C &= Z_1 \cdot B \\
D &= B^2 \cdot (C + aZ_1^2) \\
Z_3 &= C^2 \\
E &= A \cdot C \\
X_3 &= A^2 + D + E \\
F &= X_3 + X_2 \cdot Z_3 \\
G &= X_3 + Y_2 \cdot Z_3 \\
Y_3 &= E \cdot F + Z_3 \cdot G
\end{aligned}$$

0.8 Example Consider point doubling in projective coordinate system. Given an elliptic curve: $Y^2 + XYZ = X^3Z + aX^2Z^2 + bZ^4$. Let $2(X_1, Y_1, Z_1) = (X_3, Y_3, Z_3)$, then

$$\begin{aligned}
X_3 &= X_1^4 + b \cdot Z_1^4 \\
Z_3 &= X_1^2 \cdot Z_1^2 \\
Y_3 &= bZ_1^4 \cdot Z_3 + X_3 \cdot (aZ_3 + Y_1^2 + bZ_1^4)
\end{aligned}$$

In the above examples, polynomoial multiplication and squaring operations are implemented in hardware using Montgomery or Barrett reductions over finite fields $\mathbb{F}_{2^k}$. There also exist sequential multipliers over $\mathbb{F}_{2^k}$, these are described in later chapters on sequential verification.

The field size for such applications is generally very large. For example, the U.S. National Institute for Standards and Technology (NIST) recommends, for elliptic curve cryptography, fields $\mathbb{F}_{2^k}$ where the datapath size $k = 163, 233, \ldots, 571$ bits. Such large size and complicated arithmetic nature of such circuits clearly shows the complexity of the formal verification problem. Contemporary techniques lack the requisite power of abstraction to model and verify such large systems. For this reason, I propose polynomial abstractions over finite fields to model and verify such circuits using computer algebra techniques. This is the subject of subsequent chapters. Before we get to those concepts, let us study one more concept, that of polynomial functions over finite fields $\mathbb{F}_q$.

VI Polynomial Functions $f : \mathbb{F}_q \to \mathbb{F}_q$

A combinational logic circuit with k-bit inputs and k-bit outputs implements a Boolean function that is a mapping of the type $f : \mathbb{B}^k \to \mathbb{B}^k$, where $\mathbb{B} = \{0, 1\}$. Every such function can also be viewed as a mapping among 2^k elements. Therefore, these can be modeled as functions over $f : \mathbb{F}_{2^k} \to \mathbb{F}_{2^k}$ or over $f : \mathbb{Z}_{2^k} \to \mathbb{Z}_{2^k}$. We are interested in representing these functions f symbolically by means of a (word-level) polynomial $\mathcal{F}$ — i.e. as **polynomial functions** over $\mathbb{F}_{2^k}$ or $\mathbb{Z}_{2^k}$.

A function is a polynomial function if the function (mapping) can be represented by means of a polynomial. Over finite integer rings, *not every function* $f : \mathbb{Z}_n \to \mathbb{Z}_n, n \in \mathbb{N}$ can be represented as a polynomial. However, over $\mathbb{F}_q$, *every function* is a polynomial function, i.e. every function can be represented as a polynomial.

0.9 Theorem From [2]: Any function $f : \mathbb{F}_q \to \mathbb{F}_q$ is a polynomial function over $\mathbb{F}_q$, that is there exists a polynomial $\mathcal{F} \in \mathbb{F}_q[x]$ such that $f(a) = \mathcal{F}(a)$, for all $a \in \mathbb{F}_q$.

By analyzing f over each of the q points, one can apply **Lagrange's interpolation formula** and interpolate a polynomial

$$\mathcal{F}(x) = \sum_{n=1}^{q} \frac{\prod_{i \neq n}(x - x_i)}{\prod_{i \neq n}(x_n - x_i)} \cdot f(x_n), \qquad \text{(One.5)}$$

which is a polynomial of degree at most $q - 1$ in x. One can easily see that $\mathcal{F}(a) = f(a)$ for all $a \in \mathbb{F}_q$, and $\mathcal{F}(x)$ is therefore the polynomial representation of the function f.

0.10 Example *Let* $A = \{a_2, a_1, a_0\}$ *and* $Z = \{z_2, z_1, z_0\}$ *be 3-bit vectors. Consider the function* $Z[2:0] = A[2:0] >> 1$, *i.e. a bit-vector right shift operation on* A. *The function maps as follows:*

$\{a_2a_1a_0\}$	A	$\rightarrow$	$\{z_2z_1z_0\}$	Z
000	*0*	$\rightarrow$	*000*	*0*
001	*1*	$\rightarrow$	*000*	*0*
010	α	$\rightarrow$	*001*	*1*
011	$\alpha + 1$	$\rightarrow$	*001*	*1*
100	α^2	$\rightarrow$	*010*	α
101	$\alpha^2 + 1$	$\rightarrow$	*010*	α
110	$\alpha^2 + \alpha$	$\rightarrow$	*011*	$\alpha + 1$
111	$\alpha^2 + \alpha + 1$	$\rightarrow$	*011*	$\alpha + 1$

By applying Lagrange's interpolation formula over $\mathbb{F}_{2^3}$, *we obtain* $Z = (\alpha^2 + 1)A^4 + (\alpha^2 + 1)A^2$, *as the polynomial representation of the function, where* $P(\alpha) = \alpha^3 + \alpha + 1 = 0$.

We know that for all elements $A \in \mathbb{F}_q, A^q = A$, and hence $A^q - A = 0$. Therefore, the polynomial $X^q - X$ *vanishes* on all points in $\mathbb{F}_q$. Consequently, any polynomial $\mathcal{F}(X)$ can be reduced $\pmod{X^q - X}$ to obtain a *canonical* representation $(\mathcal{F}(X) \pmod{X^q - X})$ with degree at most $q - 1$.

0.11 Definition Any function $f : \mathbb{F}_q^d \rightarrow \mathbb{F}_q$ has a unique canonical representation (UCR) as a polynomial $\mathcal{F} \in \mathbb{F}_q[x_1, \ldots, x_d]$ such that all its nonzero monomials are of the form $x_1^{i_1} \cdots x_d^{i_d}$ where $0 \leqslant i_j \leqslant q - 1$, for all $j = 1, \ldots, d$.

Canonical representations of such polynomial functions are important as they allow to perform verification of GF circuits. However, the point-wise interpolation is infeasible over large fields. In later chapters, we will study how the Gröbner basis engines can be employed on circuits, using specific term orders derived from the circuit's topology, to derive such canonical polynomial representations over $\mathbb{F}_{2^k}$.

Bibliography

[1] R. J. McEliece, *Finite Fields for Computer Scientists and Engineers*. Kluwer Academic Publishers, 1987.

[2] R. Lidl and H. Niederreiter, *Finite Fields*. Cambridge University Press, 1997.

[3] *ST23YLxx series Microcontroller for Smart Cards*, ST Microelectronics.

[4] K. Kobayashi, "Studies on Hardware Assisted Implementation of Arithmetic Operations in Galois Field," Ph.D. dissertation, Nagoya University, Japan, 2009.

[5] S. Morioka and Y. Katayama, "Design methodology for a one-shot reed-solomon encoder and decoder," in *IEEE International Conference on Computer Design*, 1999, pp. 60–67.

[6] Y. Lee, K. Sakiyama, L. Batina, and I. Verbauwhede, "Elliptic-Curve-Based Security Processor for RFID," *IEEE Transactions on Computers*, vol. 57, no. 11, pp. 1514–1527, Nov. 2008. [Online]. Available: http://dx.doi.org/10.1109/TC.2008.148

[7] D. Hankerson, J. Hernandez, and A. Menezes, "Software Implementation of Elliptic Curve Cryptography over Binary Fields," 2000.

[8] E. Mastrovito, "VLSI Designs for Multiplication Over Finite Fields GF(2^m)," *Lecture Notes in CS*, vol. 357, pp. 297–309, 1989.

[9] V. Miller, "Use of Elliptic Curves in Cryptography," in *Lecture Notes in Computer Sciences*. New York, NY, USA: Springer-Verlag New York, Inc., 1986, pp. 417–426. [Online]. Available: http://dl.acm.org/citation.cfm?id=18262.25413

[10] P. Montgomery, "Modular Multiplication Without Trial Division," *Mathematics of Computation*, vol. 44, no. 170, pp. 519–521, Apr. 1985.

[11] C. Koc and T. Acar, "Montgomery Multiplication in GF(2^k)," *Designs, Codes and Cryptography*, vol. 14, no. 1, pp. 57–69, Apr. 1998.

[12] M. Knežević, K. Sakiyama, J. Fan, and I. Verbauwhede, "Modular Reduction in GF(2^n) Without Pre-Computational Phase," in *Proceedings of the International Workshop on Arithmetic of Finite Fields*, 2008, pp. 77–87.

[13] H. Wu, "Montgomery Multiplier and Squarer for a Class of Finite Fields," *IEEE Transactions On Computers*, vol. 51, no. 5, May 2002.

[14] P. Barrett, "Implementing the Rivest Shamir and Adleman Public Key Encryption Algorithm on a Standard Digital Signal Processor," in *Proceedings of Advances In Cryptology*. London, UK, UK: Springer-Verlag, 1987, pp. 311–323. [Online]. Available: http://dl.acm.org/citation.cfm?id=36664.36688

[15] J. López and R. Dahab, "Improved Algorithms for Elliptic Curve Arithmetic in GF(2^n)," in *Proceedings of the Selected Areas in Cryptography*. London, UK, UK: Springer-Verlag, 1999, pp. 201–212. [Online]. Available: http://dl.acm.org/citation.cfm?id=646554.694442