STANFORD

Supplementary Lecture 9A

Galois Fields and Arithmetic

February 3, 2026

JOHN M. CIOFFI

Hitachi Professor Emeritus (recalled) of Engineering

Instructor EE379A – Winter 2026

Announcements & Agenda

- Announcements

- Galois Field Arithmetic
- Vector spaces over Galois Fields

Galois Field Arithmetic

Appendix B.1

Finite Field Algebra (Appendix B)

- group

Definition B.1.1 [Group] *A* **group** S *is a set, with a well-defined operation for any two members of that set, call it* **addition** *and denote it by* $+$, *that satisfies the following four properties:*

1. **Closure** $\forall\ s_1, s_2 \in S$, *the sum* $s_1 + s_2 \in S$.
2. **Associative** $\forall\ s_1, s_2, s_3 \in S$, $s_1 + (s_2 + s_3) = (s_1 + s_2) + s_3$.
3. **Identity** *There exists an identity element* 0 *such that* $s + 0 = 0 + s = s$, $\forall\ s \in S$.
4. **Inverse** $\forall\ s \in S$, *there exists an inverse element* $(-s) \in S$ *such that* $s + (-s) = (-s) + s = 0$.

- ring

Definition B.1.2 [Ring] *A* **ring** R *is an Abelian group, with the additional well-defined operation for any two members of that set, call it* **multiplication** *and denote it by* $\cdot$ *(or by no operation symbol at all), that satisfies the following three properties:*

1. **Closure for multiplication** $\forall\ r_1, r_2 \in R$, *the product* $r_1 \cdot r_2 \in R$.
2. **Associative for multiplication** $\forall\ r_1, r_2, r_3 \in R$, $r_1 \cdot (r_2 \cdot r_3) = (r_1 \cdot r_2) \cdot r_3$.
3. **Distributive** $\forall\ r_1, r_2, r_3 \in R$, *we have* $r_1 \cdot (r_2 + r_3) = r_1 \cdot r_2 + r_1 \cdot r_3$ *and* $(r_1 + r_2) \cdot r_3 = r_1 \cdot r_3 + r_2 \cdot r_3$.

Finite Field Algebra (Appendix B)

- field

Definition B.1.3 **[Field]** *A* **field** F *is a ring, with the additional operation of division, the inverse operation to multiplication, denoted by* $/$. *That is for any* $f_1, f_2 \in F$, *with* $f_2 \neq 0$, *then* $f_1/f_2 = f_3 \in F$, *and* $f_3 \cdot f_2 = f_1$.

- Vector space

Definition B.1.4 **[Vector Space]** *An* n*-dimensional* **Vector Space** V *over a field* F *contains elements called vectors* $\boldsymbol{v} = [v_{n-1}, ..., v_0]$, *each of whose components* v_i $i = 0, ..., n-1$ *is itself an element in the field* F. *The vector space is closed under addition (because the field is) and also under scalar multiplication where* $f_i \cdot \boldsymbol{v} \in V$ *for any element* $f_i \in F$ *where*

$$f_i \boldsymbol{v} = [f_i \cdot v_{n-1}, ..., f_i \cdot v_0] \ . \tag{B.1}$$

The vector space captures the commutativity, associativity, zero element (vector of all zero components), and additive inverse of addition and multiplication (by scalar of each element) of the field F. *Similarly, the mulitplicative identity is the scalar* $f_i = 1$. *A set of* J *vectors is linearly independent if*

$$\sum_{j=1}^{J} f_j \cdot \boldsymbol{v}_j = 0 \tag{B.2}$$

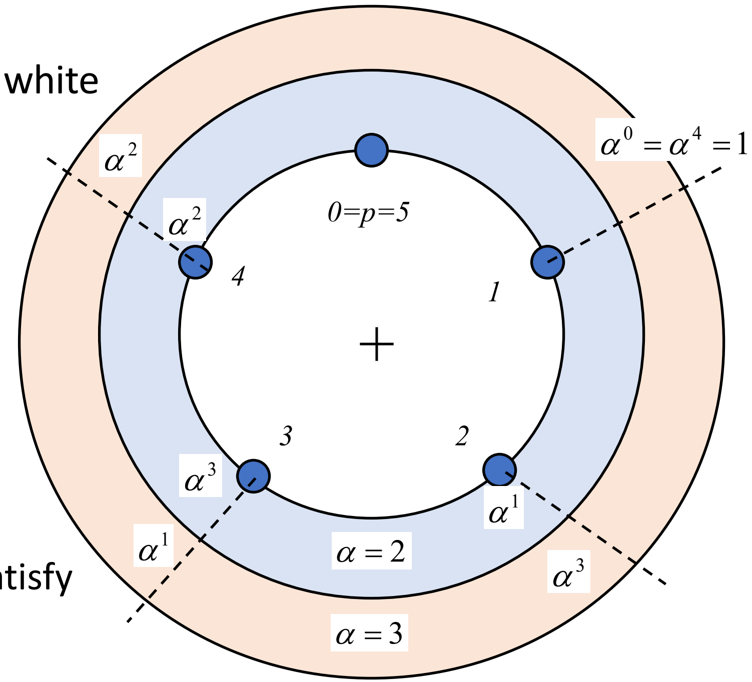
necessarily implies that

$$f_j = 0 \ \forall j \quad . \tag{B.3}$$

Galois Field for prime q

- $GF(p) = \mathbb{F}_p = \{0,1,\ldots,p-1\}$
- Adding is easy, just go around inner white circle.
  - E.g. $(2+4)_5 = 1\,; -1 = 4\,; etc.$
- Multiplication adds exponents
  - Blue or orange circles.
  - Elements $\{\alpha^0 = 1, \alpha^1, \alpha^2, \ldots, \alpha^{q-2}\}$
- For $GF(5)$, α=2 or 3 both work.
- These are **primitive elements** that satisfy
  - $1 - \alpha^4 = 1$
  - They are roots of 1 in $GF(5)$.

$$GF(5) = \{0\ \ 1\ \ \alpha\ \ \alpha^2\ \ \alpha^3\ \}$$



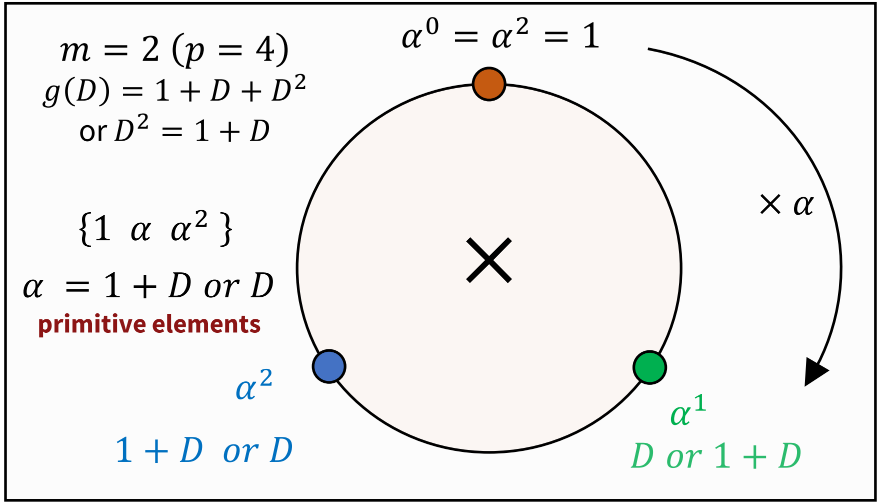
×	1	2	3	4
1	1	2	3	4
2	2	4	1	3
3	3	1	4	2
4	4	3	2	1

α	α^2	α^3	α^4
2	4	3	1
3	4	2	1

Galois Field with $p = 2^m$

- $GF(2^m) = \{0,1,\dots,2^m - 1\}$ - but elements are viewed as binary polynomials of degree m.
  - Addition/multiplication is modulo a degree-m prime **binary** polynomial.
  - $g(D) = g_0 + g_1 \cdot D + \cdots + g_{m-1} \cdot D^{m-1} + D^m$ has no $GF(2)$ factor, but it factors $D^{2^m-1} + 1 = 0$, a root of 1 **in $GF(2^m)$**.
    - This D is for a binary polynomial.

$$GF(2^m) = \{0 \;\; 1 \;\; \alpha \;\; \alpha^2 \;\; \dots \;\; \alpha^{2^m-2}\}$$



- Multiplication is modulo this prime polynomial.

- So multiply and set $g(D) = 0$

$$x(D) \cdot y(D) = d(D) \cdot g(D) + r(D)$$

$$\left(x(D) \cdot y(D)\right)_{g(D)} = r(D)$$

See example multiplication tables in Appendix B.1, as well as back-up slides

GF4 Tables ($m = 2$)

- $g(D) = 1 + D + D^2$ is a **primitive polynomial** in GF(2) of degree $m - 1$ on which GF(4) is based" $1 + D^3 = (1 + D) \cdot (1 + D + D^2)$=0
  - So, setting $g(D)$ =0 lead to $D^2 = 1 + D$ in the previous slide's example.
  - A consequent GF4 primitive element is $\alpha = D$ and $\alpha^2 = D^2 = 1 + \mathrm{D}$; or $\alpha = 1 + \mathrm{D}$ also works.

$\oplus$	0	1	D	1+D
0	0	1	D	1+D
1	1	0	1+D	D
D	D	1+D	0	1
1+D	1+D	D	1	0

$\oplus$	0	1	2	3
0	0	1	2	3
1	1	0	3	2
2	2	3	0	1
3	3	2	1	0

$GF(4) \otimes$	0	1	D	1+D
0	0	0	0	0
1	0	1	D	1+D
D	0	D	1+D	1
1+D	0	1+D	1	D

or (lsb first)

$\mathbb{F}_4 \otimes$	00	10	01	11
00	00	00	00	00
10	00	10	01	11
01	00	01	11	10
11	00	11	10	01

or (lsb last)

$\mathbb{F}_4 \otimes$	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	3	1
3	0	3	1	2

GF8 Tables: $1 + D^7 = (1 + D + D^3) \cdot (1 + D^2 + D^3) \cdot (1 + D)$ =0

$\oplus$	0	1	D	D^2	$1+D$	$D+D^2$	$1+D+D^2$	$1+D^2$
0	0	1	D	D^2	$1+D$	$D+D^2$	$1+D+D^2$	$1+D^2$
1	1	0	$1+D$	$1+D^2$	D	$1+D+D^2$	$D+D^2$	D^2
D	D	$1+D$	0	$D+D^2$	1	D^2	$1+D^2$	$1+D+D^2$
D^2	D^2	$1+D^2$	$D+D^2$	0	$1+D+D^2$	D	$1+D$	1
$1+D$	$1+D$	D	1	$1+D+D^2$	0	$1+D^2$	D^2	$D+D^2$
$D+D^2$	$D+D^2$	$1+D+D^2$	D^2	D	$1+D^2$	0	1	$1+D$
$1+D+D^2$	$1+D+D^2$	$D+D^2$	$1+D^2$	$1+D$	D^2	1	0	D
$1+D^2$	$1+D^2$	D^2	$1+D+D^2$	1	$D+D^2$	$1+D$	D	0

- $g(D) = 1 + D + D^3$, so $D^3 \rightarrow 1 + D$
  - One easy primitive element choice is $\alpha = D$.

i	GF(8) element		
	α^i	lsb first	lsb last
$-\infty$	0	000	0
0	1	100	1
1	D	010	2
2	D^2	001	4
3	$1+D$	011	6
4	$D+D^2$	110	3
5	$1+D+D^2$	111	7
6	$1+D^2$	010	5

$\oplus$	0	1	2	4	6	3	7	5
0	0	1	2	4	6	3	7	5
1	1	0	3	5	2	7	6	4
2	2	3	0	6	4	1	5	7
4	4	5	6	0	7	2	3	1
6	6	2	1	7	0	5	4	6
3	3	7	4	2	5	0	1	3
7	7	6	5	3	4	1	0	2
5	5	4	7	1	6	3	2	0

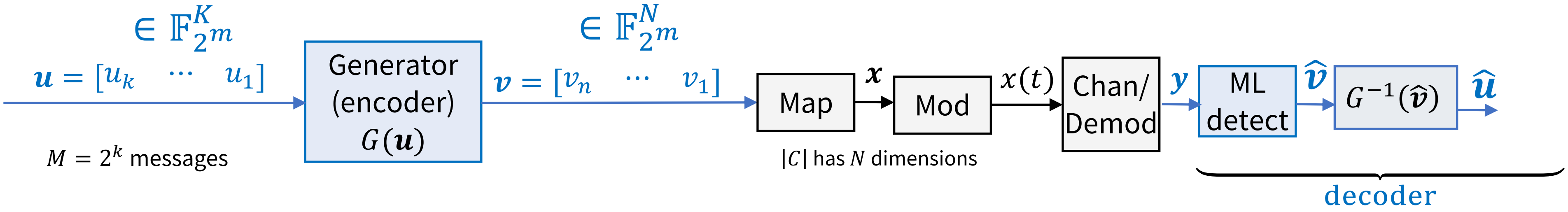
Basic logic circuits can also implement + and x ops, ("Karnaugh Maps") although look-up table is relatively small also

- See Appendix B for matlab commands that will generate these tables,

Vector Space over $GF(2^m)$

Section 7.2

$GF(2^m)$ Code Revisited



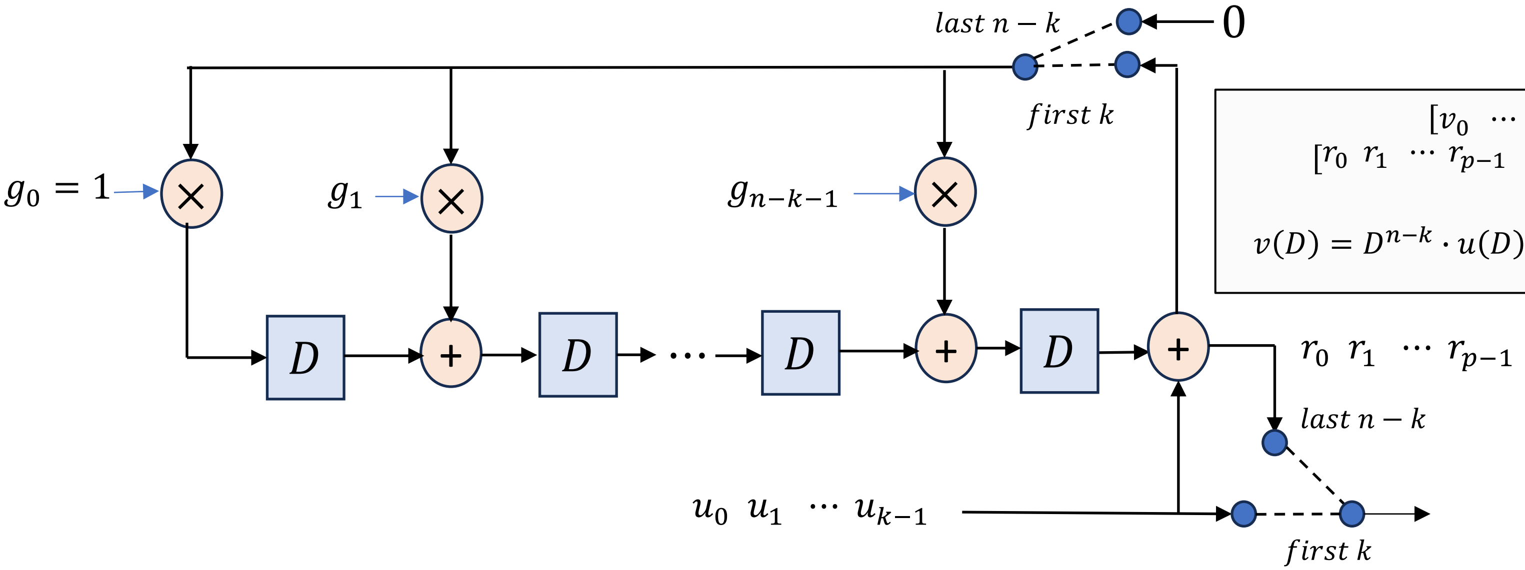
- So now each **subsymbol** is in $GF(2^m)$. So it is kind of finite-field **twice**!
- The variable D nominally is zeroed, BUT cyclic codes reintroduce it within the block.
  - Cyclic codes' codewords will all be cyclic shifts of one another. Thus, D is basically (almost) a cyclic shift.
- Very high d_{free} is possible, and ML decoders can have reasonable complexity.
- BCH Codes are cyclic; the most famous are Reed Solomon codes (attain the Singleton Bound)
  - Essentially best ball packing in the N −dimensional vector space of codewords with subsymbols in $GF(2^m)$

Conjugacy Classes

- Let q be prime and arithmetic be in $GF(q)$.
- $\alpha^p + \beta^p = (\alpha + \beta)^p$ (proof is easy, see Appendix B.1).
- **Conjugates** of α are α^{p^i} for $i = [1{:}r]$, where r is lowest $i \ni \alpha^{p^i}$=1.
- Further $\alpha^{p^i} + \beta^{p^i} = (\alpha + \beta)^{p^i}$.
- If α is root of $g(D)$, then so are its conjugates.
- **Conjugacy classes**:

$$\{\alpha, \alpha^2, \alpha^4, \alpha^8, ...\}$$
$$\{\alpha^3, \alpha^6, \alpha^{12}, \alpha^{24}, ...\}$$
$$\{\alpha^5, \alpha^{10}, \alpha^{20}, \alpha^{40}, ...\}$$
$$\{\alpha^7, \alpha^{14}, \alpha^{28}, \alpha^{56}, ...\}$$

Encoder Circuit

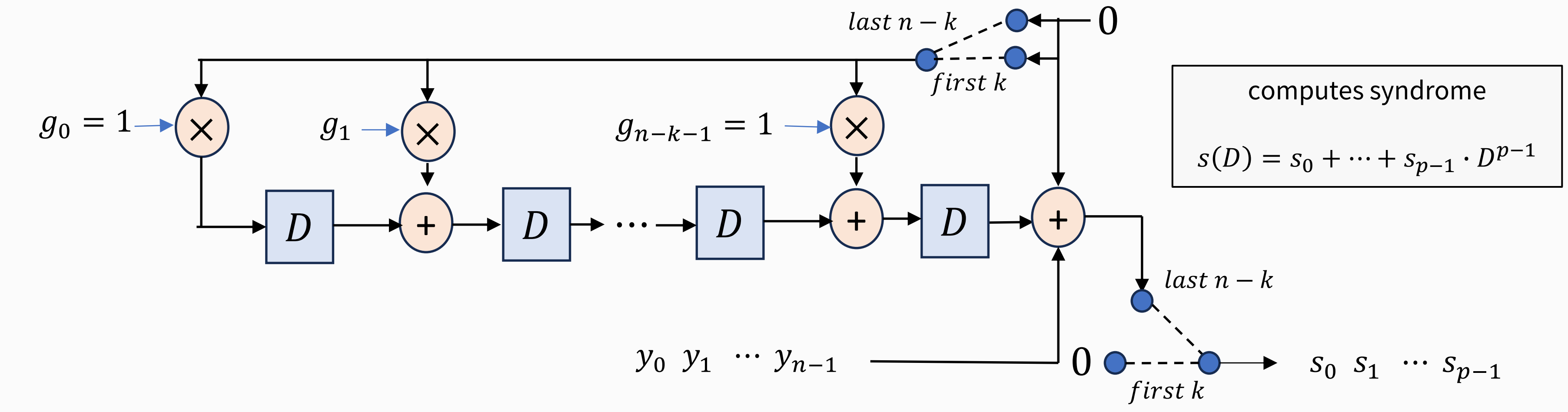


$$[v_0 \ \cdots \ v_{n-1}] = [r_0 \ r_1 \ \cdots \ r_{p-1} \ \ u_0 \ u_1 \ \cdots \ u_{k-1}]$$

$$v(D) = D^{n-k} \cdot u(D) + \left(D^{n-k} \cdot u(D)\right)_{g(D)}$$

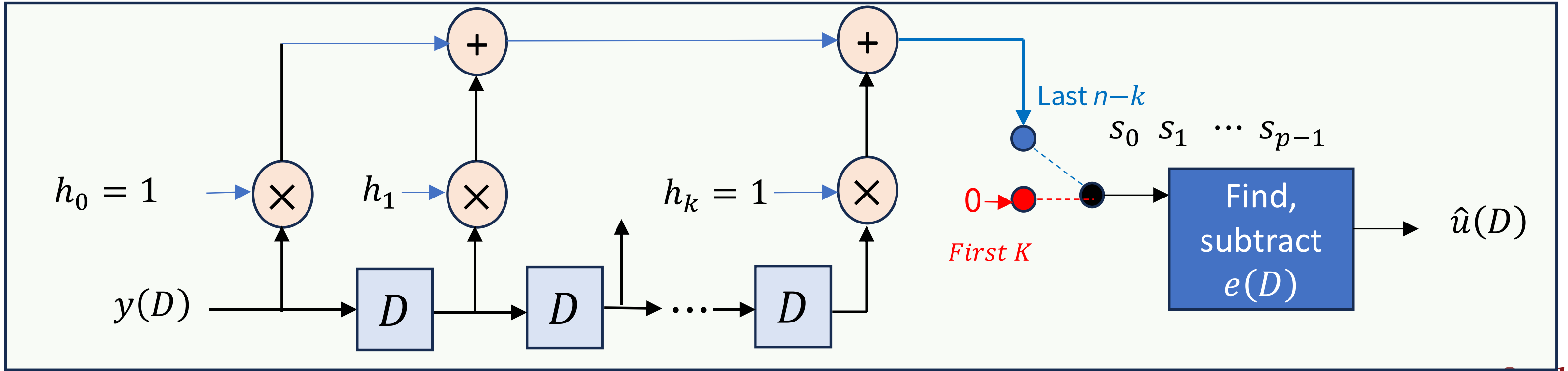
- Systematic realization (has feedback), appends the remainder in past p positions.
- Same as for binary BCH codes, but the multiplication by g_i is in $GF(2^m)$.

Syndrome Calculation



- Almost the same as encoder, except it essentially adds the remainder back (if no errors, $s(D) = 0$)

OR



Binary vs Non-Binary BCH Codes

- Binary BCH codes choose the generator $g(D)$ to be a product of primitive polynomials in $GF(2)$.
  - These are in L9:12 in main lecture, their (maximum) length is $n = 2^{m-1}$ bits
  - (Nontrivial) Binary BCH codes do not meet the Singleton Bound.
  - Their encoders and syndrome computations use binary arithmetic, but the ML decoder uses $GF(2^m)$ arithmetic.
    - Using the primitive polynomials roots in $GF(2^m)$.
- Non-Binary Reed Solomon ($\subseteq$ BCH) codes have length $N = 2^m - 1$ subsymbols, each ss in $GF(2^m)$.
  - The generator is a product of polynomials (not necessarily primitive) with coefficients in $GF(2^m)$.
  - All the arithmetic, including ML decoder, is in $GF(2^m)$.
  - See L12
- The conjugacy classes of roots in $GF(2^m)$ that all are roots of a specific primitive binary polynomial in $GF(2)$ are not necessary in the Reed Solomon codes.
- The "Y" values (error magnitudes) are easy in binary, but nontrivial in RS
  - But there are algorithms for finding these error magnitudes.
- These will be discussed in later supplementary lecture for Lecture 12.

STANFORD

End Lecture S9A