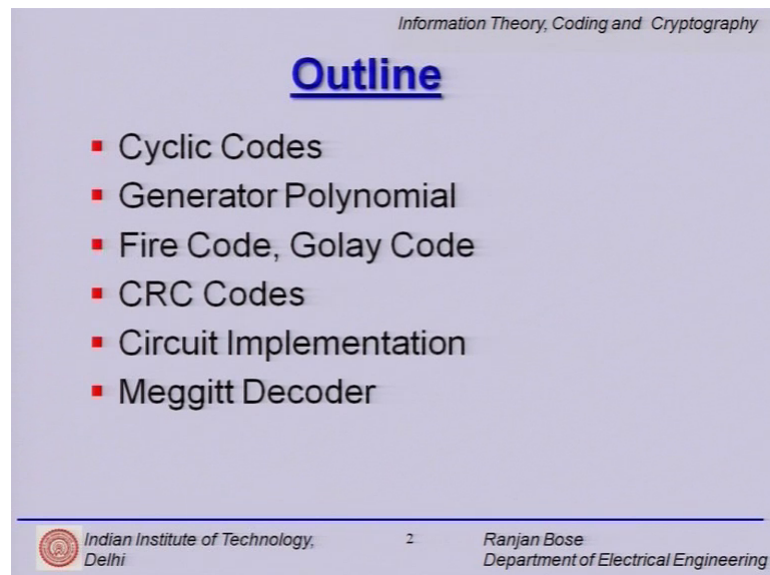


**Information Theory, Coding and Cryptography**  
**Dr. Ranjan Bose**  
**Department of Electrical Engineering**  
**Indian Institute of Technology, Delhi**

**Module - 22**  
**Cyclic Codes**  
**Lecture - 22**

Hello and welcome to our next lecture on Cyclic Codes. Let us start with a brief outline of today's talk.

(Refer Slide Time: 00:35)



Information Theory, Coding and Cryptography

## Outline

- Cyclic Codes
- Generator Polynomial
- Fire Code, Golay Code
- CRC Codes
- Circuit Implementation
- Meggitt Decoder

Indian Institute of Technology, Delhi 2 Ranjan Bose  
Department of Electrical Engineering

We will revisit the concepts of cyclic codes followed by the generator polynomial and the syndrome polynomial, then we will go on to study some of the examples of good cyclic codes the fire code, the Golay code, CRC codes which are redundancy check codes. And finally, we will see why cyclic codes are so popular because they are very very so to say friendly in terms of circuit implementation, then we will talk about the Meggitt decoder.

(Refer Slide Time: 01:09)

Information Theory, Coding and Cryptography

## Recap

- Cyclic Codes
- Generator Polynomial
- Syndrome Polynomial
- Matrix Representation

Indian Institute of Technology, Delhi 3 Ranjan Bose  
Department of Electrical Engineering

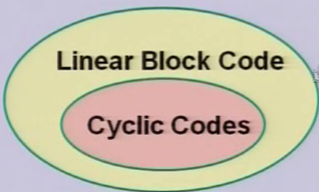
Let us quickly start with a quick recap we have already studied the basics of cyclic codes; the generator polynomial syndrome polynomial, and finally the matrix representation because cyclic codes form a subclass of linear block codes.

(Refer Slide Time: 01:26)

Information Theory, Coding and Cryptography

## Cyclic Codes

- A code  $C$  is **cyclic** if
  - $C$  is a linear code, and,
  - any **cyclic shift** of a codeword is also a codeword, i.e., if the codeword  $a_0a_1\dots a_{n-1}$  is in  $C$  then  $a_{n-1}a_0\dots a_{n-2}$  is also in  $C$ .



Indian Institute of Technology, Delhi 4 Ranjan Bose  
Department of Electrical Engineering

So, very quickly a code  $C$  is cyclic when  $C$  is a linear code and any cyclic shift of a code word also results in a valid cyclic code word. So, cyclic codes are a subclass of linear block codes we have studied all of that.


(Refer Slide Time: 01:48)

Information Theory, Coding and Cryptography

## Generator Polynomial

- Let  $C$  be a  $(n, k)$  non zero cyclic code in  $R_n$ .
- 1** ▪ Then there exists a unique monic polynomial  $g(x)$  of the smallest degree in  $C$ ,
- 2** ▪ the cyclic code  $C$  consists of all multiples of the **generator polynomial**  $g(x)$  by polynomials of degree  $k - 1$  or less.
- 3** ▪  $g(x)$  is a factor of  $x^n - 1$ .

5

 Indian Institute of Technology, DelhiRanjan Bose  
Department of Electrical Engineering

We also saw that it is possible to have one generator polynomial  $g(x)$  which can generate all the other code word polynomials before that we established a one to one link between a polynomial and a code word. What we also observed is that  $g(x)$  is necessarily a factor of  $x^n - 1$ ; where  $n$  is the code block length.

(Refer Slide Time: 02:39)

Information Theory, Coding and Cryptography

## Using $g(x)$


- A simple encoding rule to generate the codewords from the generator polynomial is

$$c(x) = i(x)g(x),$$

- where  $i(x)$  is the information polynomial,  $c(x)$  is the codeword polynomial and  $g(x)$  is the generator polynomial.
- We have seen already that there is a one to one correspondence between a word (vector) and a polynomial.
- The error vector can be also represented as the error polynomial,  $e(x)$ .
- Thus, the received word at the receiver, after passing through a noisy channel can be expressed as

$$r(x) = c(x) + e(x),$$

6

 Indian Institute of Technology, DelhiRanjan Bose  
Department of Electrical Engineering

So, these are the 3 properties that we actually observed about cyclic codes; that this  $g(x)$  is a unique monic polynomial and the multiplication of  $g(x)$  by any arbitrary polynomial of degree  $k - 1$  or less results in a valid code word. So we now look at how we use  $g(x)$

$x$  to generate a code word polynomial  $c$  of  $x$ , it is simply in the product of an information polynomial  $i$  of  $x$  degree  $k$  minus 1 or less multiplied with  $g$  of  $x$  the generator polynomial. We have already established a one to one correspondence between a vector and a polynomial therefore an error vector can equivalently be represented by an error polynomial  $e$  of  $x$ .

So, what we receive is  $nu$   $x$  equal to  $c$   $x$  plus  $e$   $x$  where  $e$   $x$  is the error polynomial.

(Refer Slide Time: 03:23)

*Information Theory, Coding and Cryptography*


## Syndrome Polynomial

- We define the **Syndrome Polynomial**,  $s(x)$  as the remainder of  $v(x)$  under division by  $g(x)$ , i.e.,

$$\begin{aligned} s(x) &= R_{g(x)}[v(x)] = R_{g(x)}[c(x) + e(x)] \\ &= R_{g(x)}[c(x)] + R_{g(x)}[e(x)] = R_{g(x)}[e(x)], \end{aligned}$$

- This is because  $R_{g(x)}[c(x)] = 0$ .

7

 Indian Institute of Technology, Delhi
Ranjan Bose  
Department of Electrical Engineering

We now go on to defining a syndrome polynomial  $s$  of  $x$  what we do is  $s$  of  $x$  is nothing but the remainder that you get when you get this  $nu$  of  $x$  divided by  $g$  of  $x$ . And therefore, we can easily see that it is nothing but the remainder what you get when you divide  $e$  of  $x$  by  $g$  of  $x$  right.



(Refer Slide Time: 03:49)

*Information Theory, Coding and Cryptography*

## Matrix Representation


- Suppose  $C$  is a cyclic code with generator polynomial  $g(x) = g_0 + g_1x + \dots + g_rx^r$  of degree  $r$ .
- Then the generator matrix of  $C$  is given by

$$G = \begin{bmatrix} g_0 & g_1 & \dots & g_r & 0 & 0 & 0 & \dots & 0 \\ 0 & g_0 & g_1 & \dots & g_r & 0 & 0 & \dots & 0 \\ 0 & 0 & g_0 & g_1 & \dots & g_r & 0 & \dots & 0 \\ \vdots & \vdots & & & & & & \vdots & \\ 0 & 0 & 0 & 0 & 0 & g_0 & g_1 & \dots & g_r \end{bmatrix}$$

$n$  columns

$k = (n - r)$  rows

---

 Indian Institute of Technology, Delhi8Ranjan Bose  
Department of Electrical Engineering

We also studied the matrix representation simply because a cyclic codes form a subclass of a linear block codes, so we can see a very very symmetric kind of representation of  $g$ ; so what does what are the rows of  $g$ , each row of  $g$  represents a valid code word

But here we have the first two as  $g_0 g_1$  up to  $g_r$  right and then followed by all 0's; the next row which is again a cyclic shift. Hence, another valid code word it is nothing but  $g_0 g_1$  shifted up to  $g_r$  and then remainder are added with 0's so and so forth till  $k$  shifts. And we have finally, the  $k$ -th row as  $g_0$ 's proceeding  $g_0$  up to  $g_r$ . So this is a very interesting structure of a generator matrix for a cyclic code.


(Refer Slide Time: 04:53)

Information Theory, Coding and Cryptography

### Example

- To find the generator matrices of **all ternary codes** (i.e., codes over  $GF(3)$ ) of blocklength  $n = 4$ , we first factorize  $x^4 - 1$ .
- $x^4 - 1 = (x - 1)(x^3 + x^2 + x + 1) = (x - 1)(x + 1)(x^2 + 1)$ .
- Thus all the divisors of  $x^4 - 1$  are:
  - 1,  $(x - 1)$ ,  $(x + 1)$ ,  $(x^2 + 1)$ ,
  - $(x - 1)(x + 1) = (x^2 - 1)$ ,
  - $(x - 1)(x^2 + 1) = (x^3 - x^2 + x + 1)$ ,
  - $(x + 1)(x^2 + 1) = (x^3 + x^2 + x + 1)$  and  $(x^4 - 1)$ .
- We know that all these factors of  $x^4 - 1$  are capable of generating a cyclic code.
- Note that  $-1 = 2$  for  $GF(3)$ .

9

 Indian Institute of Technology,  
Delhi

Ranjan Bose  
Department of Electrical Engineering

So, if you want to find out the cyclic codes it is just a matter of factorising  $x^n - 1$ .

We quickly look at an example, if you are interested in finding out all possible ternary codes what does that mean; a ternary code are codes with coefficients over  $GF(3)$  and the block length is specified as 4 then we just factorise  $x^4 - 1$  over  $GF(3)$  and we get this 3 factors, but please note any factor of  $x^n - 1$  is a valid candidate for a generator polynomial. So clearly we have  $x - 1$  and of course, we have to start with the trivial 1 as a factor 1 is a factor of all of those  $x^n - 1$ , then we have  $x - 1$   $x + 1$  as written here  $x^2 + 1$  also given here.

But then the product of first 2 terms is also a valid factor and hence a valid generator polynomial this gives us as  $x^2 - 1$  as a valid generator polynomial; then we take number 2 and number 3 multiply them out we get another and 1 and 3 and I can get another one and multiply all of them together which leads to  $x^4 - 1$  which is the factor of itself and hence these are. So we have actually to a total of 8 factors some of them are trivial, but they are all technically speaking generator polynomials which will lead ternary codes ternary cyclic codes over  $GF(3)$  of block length  $n$  is equal to 4.

Student: Excuse me sir.

Yes.

Student: Sir,  $x$  to the power 4 minus 1 it is defined over GF 3 so it should not be  $x$  to the power 4 plus 3 or plus 2 it is modulo operation so how can I factorise it.

So the question being asked is about the factorization of  $x$  raise to power 4 minus 1 over GF 3. So please note that any  $x$  raise power  $n$  minus 1 over any field can first be expressed as  $x$  minus 1 into  $x$  raise to power  $n$  minus 1 plus  $x$  raise power  $n$  minus 2 plus so and so forth up to  $x$  plus 1 this is an identity. So the first term is fixed, there is no doubt about this one; this is true for any field.

Now, we go further and we see that we are working over GF 3, so we focus our attention on this second term  $x$  cubed plus  $x$  square plus  $x$  plus 1. Now we try to see whether there is a linear factor here or not and we substitute, so  $x$  plus 1 now 1 plus 2 is equal to 0 in GF 3; so this  $x$  plus one is also equal to  $x$  minus 2 so the second term is  $x$  minus 2. So we have to substitute 2 here at  $x$  and see whether you get a 0 and if you substitute 2 here you can check out that yes  $x$  is equal to 2 is a factor here and consequently  $x$  plus 1 leads to be one of the factors of  $x$  raise power 4 minus 1. Now we divide this  $x$  cubed plus  $x$  square plus  $x$  plus 1 by  $x$  plus 1 and I can get this factor out, so we factorise the second term into  $x$  plus 1 times  $x$  square plus 1. So it works out and you can expand this out to get that this is the second term, so it is a straight forward factorization of  $x$  raise power 4 minus 1 over GF 3.

So, all of these 8 factors some are trivial some are not are capable of generating a cyclic code each; please note we have use the fact that minus 1 is plus 2 because 1 plus 2 is equal to 0 over GF 3.

(Refer Slide Time: 09:15)


*Information Theory, Coding and Cryptography*

### Example

- Cyclic codes of blocklength  $n = 4$  over  $GF(3)$ .

$g(x)$	$(n, k)$	$d^*$	$G$
1	(4, 4)	1	$[I_4]$
$(x-1)$	(4, 3)	2	$\begin{bmatrix} -1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 1 \end{bmatrix}$
$(x+1)$	(4, 3)	2	$\begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}$
$(x^2+1)$	(4, 2)	2	$\begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}$

10

 Indian Institute of Technology, Delhi
Ranjan Bose  
Department of Electrical Engineering

And we have listed out all of these 8 generator polynomials in this table this one contains first 4 and then the next table will contain the other one. So if you look at  $g(x)$  is equal to 1 whether it is a trivial  $n$  is equal to  $k$  is equal to four, this is not really going to give me any error correction and it is 1 for it consists of all the possible code all the possible vectors here and so there is really no minimum distance really useful for us.

Now, if you look at  $g(x)$  equal to  $x - 1$  right; so we are looking at the first factor  $x - 1$  here, so if you look at that then you can get the degree of  $g(x)$  is  $n - k$   $n - k$  is 1  $n$  is equal to 4 consequently  $k$  is equal to 3 and therefore, you can get a 3 cross 4 matrix as follows. Please note, since we are working over  $GF(3)$  you can either write minus 1 or substitute by 2. So it this could as well be written as the first two as 2 1 0 0 0 2 1 0 0 0 2 1 as a valid generator matrix over  $GF(3)$  and the elements of the generator matrix are coefficients taken from  $GF(3)$ .

Similarly,  $x + 1$  is different and  $x^2 + 1$  again you have  $n - k$  is equal to 2  $n$  is equal to 4 consequently  $k$  is equal to 2 and therefore, we have a parallel definition for a  $g$  here.

(Refer Slide Time: 11:05)

Information Theory, Coding and Cryptography

### Example

- Cyclic codes of blocklength  $n = 4$  over  $GF(3)$ .

$g(x)$	$(n, k)$	$d^*$	$G$
$(x^2 - 1)$	$(4, 2)$	2	$\begin{bmatrix} -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix}$
$(x^3 - x^2 + x - 1)$	$(4, 1)$	4	$[-1 \ 1 \ -1 \ 1]$
$(x^3 + x^2 + x + 1)$	$(4, 1)$	4	$[1 \ 1 \ 1 \ 1]$
$(x^4 - 1)$	$(4, 1)$	0	$[0 \ 0 \ 0 \ 0]$

11

Indian Institute of Technology,  
Delhi
Ranjan Bose  
Department of Electrical Engineering

We look at the remainder 4 factors and we can write it out as follows: so if you see we have been able to get 8 distinct generator polynomials and their equivalent generator matrices from here.

(Refer Slide Time: 11:26)

Information Theory, Coding and Cryptography

### Parity Check Polynomial

- Do we have a parity check polynomial corresponding to our generator polynomial,  $g(x)$ ?
- We already know that  $g(x)$  is a factor of  $x^n - 1$ .
- Hence we can write

$$x^n - 1 = h(x)g(x),$$

where  $h(x)$  is some polynomial.

- The following can be concluded by simply observing the above equation:
  - Since  $g(x)$  is monic,  $h(x)$  has to be monic because the left hand side of the equation is also monic (the leading coefficient is unity).
  - Since degree of  $g(x)$  is  $n - k$ , the degree of  $h(x)$  must be  $k$ .

12

Indian Institute of Technology,  
Delhi
Ranjan Bose  
Department of Electrical Engineering

Now, let us look at the notion of a parity check polynomial, please remember we had the notion of parity check matrix for linear block codes we are working in a subclass and therefore, it is a possible to find a parity check polynomials. So we start with a question do we really have a parity check polynomial corresponding to a generator polynomial  $g$

of  $x$ ; so we already know that  $g$  of  $x$  is a factor of  $x$  raise power  $n$  minus  $1$ , where  $n$  is the block length. So we can always write  $x$  raise power  $n$  minus  $1$  as some polynomial  $h$  of  $x$  into  $g$  of  $x$ , because  $g$  of  $x$  is a factor.

Now, we observe that  $g$  of  $x$  is a monic polynomial by definition consequently  $h$  of  $x$  has to be monic because the left hand side is also monic monic means leading coefficient is unity. Now degree of  $g$  of  $x$  is  $n$  minus  $k$  so little bit of observation will tell you that the degree of  $h$  of  $x$  must be  $k$ , so that their product leads to  $x$  raise  $n$  as the highest power of  $x$ .


(Refer Slide Time: 12:44)

Information Theory, Coding and Cryptography

## Parity Check Polynomial

- Suppose  $C$  is a cyclic code in  $R_n$  with the generator polynomial  $g(x)$ .
- Recall that we are denoting  $F[x]/f(x)$  by  $R_n$ , where  $f(x) = x^n - 1$ .
- In  $R_n$ ,  $h(x)g(x) = x^n - 1 = 0$ .
- Then, any codeword belonging to  $C$  can be written as  $c(x) = a(x)g(x)$ , where the polynomial  $a(x) \in R_n$ .
- Therefore in  $R_n$ ,
 
$$c(x)h(x) = a(x)g(x)h(x) = c(x) \cdot 0 = 0.$$
- Thus,  $h(x)$  behaves like a **Parity Check Polynomial**.
- Any valid codeword when multiplied by the parity check polynomial yields the zero polynomial.
- This concept is parallel to that of the parity check matrix introduced in the previous chapter.
- Since we are still in the domain of linear block codes, we go ahead and define the parity check matrix in relation to the parity check polynomial.

---



Indian Institute of Technology,  
Delhi

13

Ranjan Bose  
Department of Electrical Engineering

So, now what we do is we started with a cyclic code  $C$  in  $R_n$  with the generator polynomial  $g$  of  $x$  and we are denoting  $F[x]$  set of all possible polynomial is divided by this  $f(x)$  which is the prime polynomial by  $R_n$  right and here you have  $h(x)$  into  $g(x)$  is equal to  $x$  raise power  $n$  minus  $1$ , but in  $R_n$  so when you take modulo  $x$  raise power  $n$  minus  $1$  you get it as  $0$ . So you have this  $c(x)$  which is a valid code word polynomial and if you multiply with  $h(x)$  you get  $a(x)$  into  $g(x)$  that is how you generate  $c(x)$  times  $h(x)$  right in  $R_n$  some modulo  $x$  raise power  $n$  minus  $1$ , but here we will get this equal to  $0$ .

So, consequently  $h(x)$  behaves like a parity check polynomial; what does it mean? Take any received vector and if it is a valid code word polynomial then you multiply it with  $h(x)$  and take modulo  $x$  raise power  $n$  minus  $1$ , if you get a  $0$  you can be rest assured that



it is a valid code word polynomial therefore, it behaves like a parity check polynomial. So valid code words when multiplied with  $h$  of  $x$  gives a 0, whereas if it is not then you have some errors; error detection is extremely easy with this we will explore this concept shortly.

(Refer Slide Time: 14:40)

Information Theory, Coding and Cryptography


## Parity Check Matrix

- Suppose  $\mathbf{C}$  is a cyclic code with the parity check polynomial then the parity check matrix of  $\mathbf{C}$  is given by  $h(x) = h_0 + h_1x + \dots + h_kx^k$ ,

$$H = \begin{bmatrix} h_k & h_{k-1} & \dots & h_0 & 0 & 0 & 0 & \dots & 0 \\ 0 & h_k & h_{k-1} & \dots & h_0 & 0 & 0 & \dots & 0 \\ 0 & 0 & h_k & h_{k-1} & \dots & h_0 & 0 & \dots & 0 \\ \vdots & \vdots & & & & & & \vdots & \\ 0 & 0 & 0 & 0 & 0 & h_k & h_{k-1} & \dots & h_0 \end{bmatrix} \begin{matrix} \uparrow \\ \uparrow \\ \uparrow \\ \vdots \\ \uparrow \end{matrix} \begin{matrix} (n-k) \text{ rows} \\ \\ \\ \\ \end{matrix}$$

$\leftarrow n \text{ columns} \rightarrow$

---

 Indian Institute of Technology, Delhi
14
Ranjan Bose  
Department of Electrical Engineering

Now, since we had defined a parity check polynomial we also have a parity check matrix definition. So it is very easy to define a matrix once we have the parity check polynomial please note the sequence of steps, we have  $x$  raise  $n$  minus 1 and  $x$  raise power  $n$  minus 1 can be factorized to give you  $h$  of  $x$  and  $g$  of  $x$ . And then  $h$  of  $x$  is the coefficient can easily be written in this form. So if you consider little bit more graphically then you have if you look at your slide so we have this  $x$  raise power  $n$  minus 1.

(Refer Slide Time: 15:37)

The image shows a handwritten derivation on a grid background. At the top, it states  $(x^n - 1) = h(x) \cdot g(x)$  in  $GF(q)$ . Below this, the polynomial  $h(x)$  is written as  $h_0 + h_1x + h_2x^2 + \dots + h_kx^k$  and  $g(x)$  as  $g_0 + g_1x + \dots + g_{n-k}x^{n-k}$ . Two double arrows point down to the matrices  $H$  and  $G$ . The matrix  $H$  is a  $(k+1) \times n$  matrix with the first row  $[h_k \ h_{k-1} \ \dots \ h_0 \ 0 \ \dots \ 0]$  and subsequent rows shifted to the right. The matrix  $G$  is a  $(n-k) \times n$  matrix with the first row  $[g_0 \ g_1 \ \dots \ g_{n-k} \ 0 \ \dots \ 0]$  and subsequent rows shifted to the right.

And the first step is you factorise it into  $h$  of  $x$  into  $g$  of  $x$  ok. Now this leads to your coefficients  $h_0$  plus  $h_1x$  plus  $h_2x^2$  plus  $h_kx^k$  and this  $g$  itself would lead to your coefficients  $g_0$  plus  $g_1x$  plus. Now immediately you can take this coefficients and write out your  $h$  matrix as  $h_k \ h_{k-1} \ \dots \ h_0 \ 0 \ 0 \ 0 \ 0$ .

And similarly  $0 \ h_k \ h_{k-1} \ \dots \ h_0$  then you have this  $h_k$  up to  $h_0$ . And similarly you have a  $G$  matrix which you can write  $g_0 \ g_1 \ \dots \ g_{n-k} \ 0 \ 0$  and then a right shift. So the point is your starting point is  $x$  raise to power  $n-1$  for which once you know  $n$  and you know the Galois field you are ready to factorise; once you factorise you get a great deal because you get  $g(x)$  and you get  $h(x)$ .


Here, please remember  $h$  of  $x$  is also a factor so it is another generator polynomial and for this  $g(x)$  serves as the parity check polynomial, so it is kind of a dual here and once you get these coefficients you can immediately write out  $h$  and  $g$ . So it is a very easy way to get a generator matrix and parity check matrix in the case of a cyclic code and that is why we spend enough time learning how to factorise over any arbitrary Galois field so we come back to our slides.

(Refer Slide Time: 18:44)

*Information Theory, Coding and Cryptography*

## Fire Code

- A **Fire code** is a cyclic burst error correcting code over  $GF(q)$  with the generator polynomial


$$g(x) = (x^{2^t-1}-1)p(x),$$


where  $p(x)$  is a prime polynomial over  $GF(q)$  whose degree  $m$  is not smaller than  $t$  and  $p(x)$  does not divide  $x^{2^t-1}$ .

The blocklength of the Fire code is the smallest integer  $n$  such that  $g(x)$  divides  $x^n-1$ .

A Fire code can correct all burst errors of length  $t$  or less.

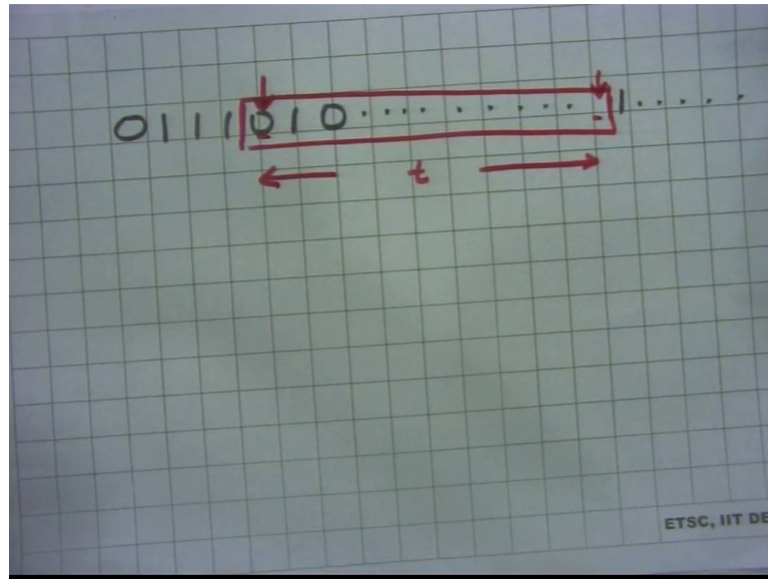
---

 *Indian Institute of Technology, Delhi* 15 *Ranjan Bose*  
Department of Electrical Engineering

And now having developed this notion of cyclic codes let us look at some examples here, we start off with a very very interesting example of a fire code. Now one of the very good selling features of cyclic code codes is the burst error correction; so a fire code is a cyclic burst error correcting code over  $GF(q)$  with the generator polynomial given as  $g(x)$  is equal to  $(x^{2^t-1}-1)p(x)$ . What is a  $p(x)$ ; where  $p(x)$  is a prime polynomial over  $GF(q)$  whose degree  $m$  is not smaller than  $t$  and  $p(x)$  does not divide  $x^{2^t-1}$  so this is relative prime to  $p(x)$ .

The block length of fire code is the smallest integer  $n$  such that  $g(x)$  divides  $x^n-1$  so you have to construct this fire code this is a mechanism how to construct it. So what is the beauty is a fire code can correct all burst errors of length  $t$  or length  $t$  or less. So a burst of error is of length  $t$  means that the first and the  $t$ -th error bits are definitely in error and the remainder may or may not be in error. So if you again look at a description here and we now define burst errors.

(Refer Slide Time: 20:37)



Suppose I have a bit stream a long enough bit stream, we take a sequence of bits and if this is  $t$  then this is a burst of error if the first one and the last one are definitely in errors and intermediate bits may or may not be in error, but this is defined as a burst error of length  $t$ . So all of them can be in error so your code should be able to correct that or detect that if it is designed to do so or maybe the alternate bits are in error, but the stretch is over  $t$  and necessarily the first one and the last one must be in error; this is how we define a burst of length  $t$ .


(Refer Slide Time: 21:47)

*Information Theory, Coding and Cryptography*

### Example

- Consider the Fire code with  $t = m = 3$ . A prime polynomial over  $GF(2)$  of degree 3 is  $p(x) = x^3 + x + 1$ , which does not divide  $(x^5 - 1)$ .
- The generator polynomial of the Fire code is
$$g(x) = (x^5 - 1)p(x) = (x^5 - 1)(x^3 + x + 1)$$
$$= x^8 + x^6 + x^5 - x^3 - x - 1$$
$$= x^8 + x^6 + x^5 + x^3 + x + 1$$
- The degree of  $g(x) = n - k = 8$ .
- The blocklength is the smallest integer  $n$  such that  $g(x)$  divides  $x^n - 1$ . After trial and error we get  $n = 35$ .
- Thus, the parameters of the Fire code are  $(35, 27)$  with
- $g(x) = x^8 + x^6 + x^5 + x^3 + x + 1$ .
- This code can correct up to bursts of length 3.
- The code rate of this code is 0.77, and is more efficient than the code generated by  $g(x) = x^6 + x^3 + x^2 + x + 1$  which has a code rate of only 0.6.

---

 Indian Institute of Technology, Delhi16Ranjan Bose  
Department of Electrical Engineering

So we come back towards slides and we continue with our fire code with  $t$  is equal to 3 is equal to  $m$  and a prime polynomial over  $GF(2)$  of degree 3 can be written as  $p(x)$  is equal to  $x^3 + x + 1$  you can check whether it is a prime polynomial you substitute  $x$  equal to 1 you do not get a 0 you substitute  $x$  is equal to 0 you do not get a 0. So there no linear factors of the type  $x$  and  $x - 1$  and then you can verify that this is indeed a prime polynomial.

Now, in our construction in the previous one  $x^{2^{t-1}} - 1$  we have already figured out what is this  $p(x)$  is  $x^3 + x + 1$  now  $2^{t-1}$  right. So we would like to correct burst up to length 3, so  $3 \leq 2^{t-1} - 1$ . So  $x^{2^{t-1}} - 1$  into  $p(x)$  is written as this so this is immediately are generator polynomial for the given fire code. Now, this generator polynomial should be able to correct burst up to length 3;  $g(x)$  has degree  $n - k$  is equal to 8 the block length is a smallest integer  $n$  such that  $g(x)$  divides  $x^{n-1} - 1$ .

Because it needs to be a cyclic code, so we try make a few attempts and we get  $n$  is equal to 35. So  $n - k$  is 8  $n$  is 35 consequently  $k$  is 27; so we find our fire code of size 35 comma 27 with the generator polynomial given as follows. Now you can check that this code rate for this fire code is 0.77 and it is more efficient than another cyclic code which has a code rate of 0.6 for examples. So fire codes are pretty efficient and as we increase  $t$  their efficiency tends to 1. So we will get better and better fire code in terms of the code rate if you have  $t$  which is the number of burst errors it can correct larger.

(Refer Slide Time: 24:25)

*Information Theory, Coding and Cryptography*

## Binary Golay Code

**For perfect codes:**


$$M \left\{ \binom{n}{0} + \binom{n}{1}(q-1) + \binom{n}{2}(q-1)^2 + \dots + \binom{n}{t}(q-1)^t \right\} = q^n$$

which is satisfied for the values:  $n = 23$ ,  $k = 12$ ,  $M = 2^k = 2^{12}$ ,  $q = 2$  and  $t = (2^k - 1)/2 = 3$ .

- **(23, 12) perfect code is the binary Golay code.**
- We shall now explore this perfect code as a cyclic code.
- We start with the factorization of  $(x^{23}-1)$ .

▾

---

 Indian Institute of Technology,  
Delhi17Ranjan Bose  
Department of Electrical Engineering

Now, we look at another example the Golay code, we start up with refreshing our definition of perfect codes if you recall for perfect codes we have this condition so we wanted the equality for the hamming bound  $n$  is the block length  $q$  is the Galois field  $GF q$  and  $n$  is block length  $t$  is a number of errors it can correct. So if you try out with different integer values of  $m$   $n$   $t$  and  $q$ . And since we are looking at binary  $q$  is equal to 2 we would get  $n$  equal to 23,  $k$  is equal to 12,  $m$  is equal to 2 raise power 12 and  $t$  is equal to 3 satisfying this condition. So maybe there is a possibility that  $n$  is equal to 23,  $k$  is equal to 12 so 23 comma 12 code exists. So Golay was to was the first one to find this and 23 comma 12 perfect code exists and this is the called the binary Golay code.

Now, we will explore this perfect code as a cyclic code so the first step is the moment I get a hang on  $n$   $n$  is equal to 23 I quickly write  $x$  raise power 23 minus 1 and try to factorise it, where over  $GF 2$  because it is a binary code.




(Refer Slide Time: 26:03)

*Information Theory, Coding and Cryptography*

## Binary Golay Code

$$\begin{aligned}(x^{23}-1) &= \\ &= (x-1) \times \\ &\quad (x^{11} + x^{10} + x^6 + x^5 + x^4 + x^2 + 1) \times \\ &\quad (x^{11} + x^9 + x^7 + x^6 + x^5 + x + 1) \\ &= (x-1)g_1(x)g_2(x).\end{aligned}$$

18

 *Indian Institute of Technology,  
Delhi* *Ranjan Bose  
Department of Electrical Engineering*

So, let us try to factorise this  $x$  raise power 23 minus 1 and with a little bit of effort you can write out these factors. So immediately we have this as a possible generator polynomial and other possible generator polynomial yet another possible generator polynomial, while this one is a trivial. And these two are of interest. So  $n$  minus  $k$  here is 11 so if you go back we should be able to get  $n$  minus  $k$  as 11 here so indeed we have hope to get a perfect cyclic code which is the binary Golay code.

So, please note that for a non-trivial scenarios  $g_1(x)$  is the first polynomial and  $g_2(x)$  is a second polynomial both are the generator polynomials of the binary Golay code 23 comma 12.

(Refer Slide Time: 27:11)


*Information Theory, Coding and Cryptography*

## Binary Golay Code

$$g(x) = (x^{11} + x^{10} + x^6 + x^5 + x^4 + x^2 + 1)$$

- The degree of  $g_1(x) = n - k = 11$ , hence  $k = 12$ , which implies that there exists a  $(23, 12)$  cyclic code.
- In order to prove that it is a perfect code, we must show that the minimum distance of this  $(23, 12)$  cyclic code is 7.
- One way is to write out the parity check matrix,  $H$ , and show that no six columns are linearly dependent.
- The code rate is  $0.52$  and it is a triple error correcting code.
- However, the relatively small block length of this perfect code makes it impractical for most real life applications.

---

 Indian Institute of Technology,  
Delhi19Ranjan Bose  
Department of Electrical Engineering

So, we start with this anyone  $g$  of  $x$ , so degree is 11  $k$  is 12 so we have found out a cyclic code which is also a perfect cyclic code, right.

Now, in order to prove that it is a perfect code well we should also show that the minimum distance is 7, why because  $t$  if you see in our previous slide we wanted  $t$  equal to 3 it should be able to correct 3 errors and we have got  $n$  23,  $k$  12, but  $d$  needs to be the  $t$  needs to be shown equal to 3. So one way is to write out the parity check matrix and show that no 6 columns are linearly independent for this you have to employ a simple computer program and you can prove that the code rate is 0.52 and this is an indeed triple error correcting code.

But this relatively small block length of this perfect code practically makes it not so practical for a real life operations code rate is also pretty poor. So even though it is a perfect code rate wire it is almost 50 percent of the bits are over head bits.

(Refer Slide Time: 28:47)


Information Theory, Coding and Cryptography

## Ternary Golay Code

$$(x^{11}-1) = (x-1)(x^5 + x^4 + x^3 + x^2 - 1)(x^5 - x^3 - x^2 - x - 1)$$
$$= (x-1)g_1(x)g_2(x).$$

- The degree of  $g_1(x) = n - k = 5$ , hence  $k = 6$ , which implies that there exists a (11, 6) cyclic code.
- In order to prove that it is a perfect code, we must show that the minimum distance of this (11, 6) cyclic code is 5.

20

 Indian Institute of Technology, Delhi Ranjan Bose  
Department of Electrical Engineering

We also have a ternary Golay code so if you have  $x$  raise power 11 minus 1 and you try to factorise you get these two factors  $n - k$  is 5 and  $k$  gives  $k$  equal to 6. So we have an 11 comma 6 ternary cyclic code which is the ternary Golay code the minimum distance of this code is 5.


(Refer Slide Time: 29:15)

Information Theory, Coding and Cryptography

## CRC Codes

- One of the common error detecting codes is the Cyclic Redundancy Check (CRC) codes.
- For a  $k$ -bit block of bits the  $(n, k)$  CRC encoder generates  $(n - k)$  bit long frame check sequence (FCS). Let us define the following
  - $T = n$ -bit frame to be transmitted
  - $D = k$ -bit message block (information bits)
  - $F = (n - k)$  bit FCS, the last  $(n - k)$  bits of  $T$
  - $P =$  the predetermined divisor, a pattern of  $(n - k + 1)$  bits.

21

 Indian Institute of Technology, Delhi Ranjan Bose  
Department of Electrical Engineering

Now, let us look at another class of codes which are called the cyclic redundancy check codes also be if you called as CRC codes; very practical very useful touches our lives on a day to day basis ok. This employees the property that cyclic codes can detect errors

very easily, so these are check codes these detect they are not in the business of correcting the codes ok. So we can have a k bit block of bits and the n comma k CRC encoder generates n minus k bit long frame check sequence.

So, you have the first k bits as the information bits and then n minus k the over head bits are the frame check sequence bits.

(Refer Slide Time: 30:16)

The slide is titled "CRC Codes" and is part of a presentation on "Information Theory, Coding and Cryptography". It contains the following text:

$$V(x) = T(x) + E(x),$$

where  $E(x)$  is the error polynomial.

- Then  $[T(x) + E(x)]/P(x) = E(x)/P(x)$  because  $T(x)/P(x) = 0$ .
- Those errors that happen to correspond to polynomials containing  $P(x)$  as a factor will slip by, and the others will be caught in the net of the CRC decoder.
- The polynomial  $P(x)$  is also called the generator polynomial for the CRC code.
- CRC codes are also known as **polynomial codes**.

At the bottom of the slide, there is a footer with the Indian Institute of Technology Delhi logo, the text "Indian Institute of Technology, Delhi", the page number "22", and the name "Ranjan Bose, Department of Electrical Engineering".

And please note as we have discussed before what we received in this case suppose  $V$  of  $x$  is nothing but what you sent  $T$  of  $x$  plus an error  $E$  of  $x$  so  $E$  of  $x$  is the error polynomial and the way to find out is we divide with using the generator polynomial  $p$  of  $x$ , but what we get is  $t$  of  $x$  is perfectly divisible by  $P$  of  $x$  we have done this before leading us to have only  $E$  of  $x$  by  $P$  of  $x$ .

Now we can design a generator polynomial smartly that it catches lot of errors. In fact, if the even number of terms in  $P$  of  $x$  then odd number of terms in  $E$  of  $x$  can be caught and so and so forth. So the polynomial  $P$  of  $x$  is called the generator polynomial for the CRC codes and CRC codes are also generally known as polynomial codes.

(Refer Slide Time: 31:14)

*Information Theory, Coding and Cryptography*


## CRC Codes

Four versions of  $P(x)$  have become international standards:

- **CRC-12:**  $P(x) = x^{12} + x^{11} + x^3 + x^2 + x^1 + 1$ .
- **CRC-16:**  $P(x) = x^{16} + x^{15} + x^2 + 1$ .
- **CRC-CCITT:**  $P(x) = x^{16} + x^{15} + x^5 + 1$ .
- **CRC-32:**  $P(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x^1 + 1$ .

- Both CRC-16 and CRC-CCITT are popular for 8-bit characters.
- They result in a 16 bit FCS and can catch all single and double errors, all errors with odd number of bits, all burst errors of length 16 or less, 99.997% of 17-bit bursts and 99.998% of 18-bit and longer bursts.
- CRC-32 is specified as an option in some point-to-point **Synchronous Transmission Standards**.

---

 *Indian Institute of Technology, Delhi* 23 *Ranjan Bose*  
*Department of Electrical Engineering*

Some of the popular CRC codes have entered into international standards and here we have listed 4 popular CRC codes both CRC 16 and CRC CCITT are popular codes for 8 bit characters and.

For example 16 bit FCS can catch all single and double errors and all errors with odd number of bits all burst errors of length 16 or less. So they are very very effective and catching burst errors and so and so forth I mean this is just the properties of these polynomials how they divide and what error patterns they can catch. CRC 32 is a part of the STS the synchronous transmission standards. So the point is these CRC code which are the subclass of cyclic codes are very practical and very powerful in correcting burst errors well they can detect definitely and some of them can correct burst errors.


(Refer Slide Time: 32:20)

*Information Theory, Coding and Cryptography*

## Circuit Implementation

- Shift registers can be used to encode and decode cyclic codes easily.
- Encoding and decoding of cyclic codes require multiplication and division by polynomials.
- The shift property of shift registers are ideally suited for such operations.
- Shift registers are banks of memory units which are capable of shifting the contents of one unit to the next at every clock pulse.
- We will focus on circuit implementation for codes over  $GF(2^m)$ .

24

 *Indian Institute of Technology,  
Delhi* *Ranjan Bose  
Department of Electrical Engineering*

Now we come to a very very important selling future of cyclic code which is that they are very amenable to circuit implementation.

So, shift registers can be used very easily to encode and decode cyclic codes ok. So what is encoding and decoding of cyclic codes well we already have a polynomial representation all we have to know is how to represent polynomials and put them in shift registers and then how do we multiply which actually comes from the FIR circuit theory. So the shift property of the shift registers can easily convert one code word into next simply because shifting right shift cyclic shift of a valid code word is another valid code word. So cyclic shifts are very very easily implemented in circuits and therefore, cyclic codes are very comfortably represented using shift registers. So we will focus and implementations for codes over  $GF(2^m)$  so you can have binary and even non binary representations.




(Refer Slide Time: 33:40)

Information Theory, Coding and Cryptography

## Circuit Implementation

- Beside the shift register, we will make use of the following circuit elements:
  - A scaler, whose job is to multiply the input by a fixed field element.
  - An adder, which takes in two inputs and adds them together. A simple circuit realization of an adder is the 'exclusive-or' or the 'xor' gate.
  - A multiplier, which is basically the 'and' gate.

25

 Indian Institute of Technology, DelhiRanjan Bose  
Department of Electrical Engineering

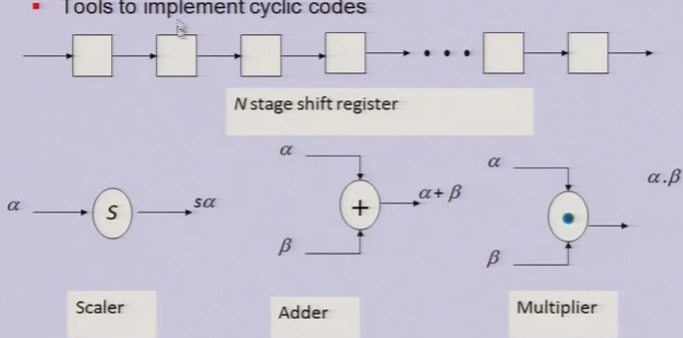
So, beside the shift register part we sometimes require for a non binary case a multiplication by a scalar, addition and multiply. So addition is nothing but a binary addition and multiplication is nothing but an and gate so circuit implementation is really the key to fast, cheap, efficient implementation of large cyclic codes.

(Refer Slide Time: 34:11)


Information Theory, Coding and Cryptography

## Circuit Implementation

- Tools to implement cyclic codes



26

 Indian Institute of Technology, DelhiRanjan Bose  
Department of Electrical Engineering

So, here are some of the tools that we use the circuit hardware tools, so you have the n stage shift register, we have a scalar multiplier, we have a simple adder and a

multiplication of 2 bits for example, so we have all of these vary fast, cheap hardware available in today's technology and therefore, cyclic codes are exceedingly important.


(Refer Slide Time: 34:43)

Information Theory, Coding and Cryptography

## Circuit Implementation

- A field element of  $GF(2)$  can simply be represented by a single bit. For  $GF(2^m)$  we require  $m$  bits to represent one element.
- For example, elements of  $GF(8)$  can be represented as the elements of the set {000, 001, 010, 011, 100, 101, 110, 111}.
- For such a representation we need three clock pulses to shift the element from one stage of the shift register to the next.

---

 Indian Institute of Technology, Delhi
27
Ranjan Bose  
Department of Electrical Engineering

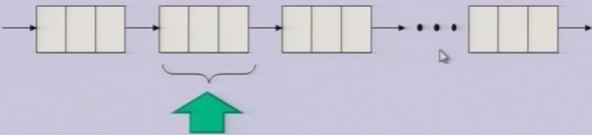
So, we take an example of how  $GF(2)$  can be used to represent  $GF(2^m)$  for example, if you have to represent  $GF(8)$  so cyclic code non binary cyclic code over  $GF(8)$  then the elements are simply represented and in their binary form. So these are the 8 elements of  $GF(8)$ , but they are easily represented in binary.

(Refer Slide Time: 35:13)

Information Theory, Coding and Cryptography


## Circuit Implementation

- Shift registers can be used to encode and decode cyclic codes easily.
- **Example:** The effective shift register for  $GF(8)$ . Any arbitrary element of this field can be represented by  $ax^2 + bx + c$ , where  $a, b, c$  are binary, and the power of the indeterminate  $x$  is used to denote the position. For example,  $101 = x^2 + 1$ .



One stage of the effective shift register

---

 Indian Institute of Technology, Delhi
28
Ranjan Bose  
Department of Electrical Engineering

So, for example, if you have to represent 1 0 1 which is nothing but a polynomial  $x^2 + 1$  why because coefficients of  $x^2$  is 1 coefficient of  $x$  is 0 missing so  $x$  is missing and coefficient of  $x^0$  is 1 is here. So if I substitute 1 0 1 here then this represents  $x^2 + 1$ , so I can represent any polynomial using the coefficients in a shift register.

(Refer Slide Time: 35:50)

*Information Theory, Coding and Cryptography*

## Circuit Implementation

Multiplication of an arbitrary field element  $(ax^2 + bx + c)$  by the element  $(x^2 + x)$  over  $GF(8)$ .

$$(ax^2 + bx + c)(x^2 + x) = ax^4 + (a+b)x^3 + (b+c)x^2 + cx$$

(modulo  $p(x)$ )

$$= (a+b+c)x^2 + (b+c)x + (a+b)$$

Indian Institute of Technology, Delhi 29 Ranjan Bose  
Department of Electrical Engineering

Now, the other thing that is required in circuit implementation of cyclic codes is the product of 2 polynomials. So far we are very comfortable representing a polynomial as a vector and vector as a polynomial there is a one to one correspondence. Now suppose we want to multiply 2 polynomials using circuits how do we do it; so my candidate polynomials are  $ax^2 + bx + c$  multiplied by for example,  $x^2 + x$  and then we have to take it modulo some  $p(x)$  so you can multiply it out and you can get these coefficients.

And if you simplify it you can write it as follows and if I would implement this using a circuit. Well, so I will take  $a$ ,  $b$  and  $c$  as coefficients here and I can always make an observation here and say ok. Let us say the first this should be my answer how do I work it the answer should have a coefficient for  $x^2$ , a coefficient for  $x$  and a coefficient for  $x^0$ . So let us first focus on the coefficient for  $x^2$ , so it should be  $a + b + c$ , so this arrow should have  $a + b + c$ .

So, these are just standard hardware implementation, so this shift register tap from here I tap from here and I tap from here gives the first coefficient. Now this one should correspond to the coefficient for  $x$ , but they should be  $b$  plus  $c$  that is very easy to implement I take a tap from  $b$   $c$  put an adder and put it and put an arrow here and same for a plus  $b$ ; so I have an  $a$  plus  $b$  here. So this simple circuit in an instant multiplies this polynomial. So it shows how easy and efficient it is to carry out, but we must note that our understanding is that this is the coefficient for  $x$  square the coefficient for  $x$  this is the coefficient for  $x$  raise power 0 same with here.

(Refer Slide Time: 38:16)

*Information Theory, Coding and Cryptography*

## Circuit Implementation

- Product of two polynomials  $b(x) = a(x)g(x)$ .

$$g(x) = g_L x^L + \dots + g_1 x + g_0, \quad a(x) = a_k x^k + \dots + a_1 x + a_0,$$

$$b(x) = a(x)g(x) \quad b(x) = b_{k+L} x^{k+L} + \dots + b_1 x + b_0.$$

*Indian Institute of Technology, Delhi*
30
*Ranjan Bose*  
Department of Electrical Engineering

So, if you have to multiply 2 polynomials in general one of them is a generator polynomial and other one is the information polynomial to get the code word polynomial and now we are really in business because this is exactly how circuit implementation for cyclic codes must work. So  $b$  of  $x$  is nothing but  $a$   $x$  into  $g$  of  $x$  and you multiply it out and you get this coefficients and then you can quickly work together to put any  $a$  of  $x$  comes in we loaded into your shift register.

So, this is the incoming vector which is represented as a polynomial or the vectors the elements are loaded here, I already have the coefficients quickly they multiply and instantaneously you get a  $b$   $x$  at the speed at which the circuit works. So you load in your information word you get the symbol out so it is really a real time operation.

(Refer Slide Time: 39:22)

*Information Theory, Coding and Cryptography*

## Dividing with $g(x)$

- The division process can be expressed as a pair of recursive equations.
- Let  $Q^{(r)}(x)$  and  $R^{(r)}(x)$  be the quotient polynomial and the remainder polynomial at the  $r^{\text{th}}$  recursion step, with the initial conditions  $Q^{(0)}(x) = 0$  and  $R^{(0)}(x) = a(x)$ .

$$Q^{(r)}(x) = Q^{(r-1)}(x) + R_{(n-r)}x^{k-r},$$

$$R^{(r)}(x) = R^{(r-1)}(x) - R_{(n-r)}x^{k-r}g(x),$$

31

Indian Institute of Technology,  
Delhi
Ranjan Bose  
Department of Electrical Engineering

The other part of good cyclic code is the division part we need to divide to carry out this checks CRC does that all other cyclic code error detection routinely involves dividing by  $g$  of  $x$  or some polynomial. So division process can also be expressed as a pair of recursive equations, so let  $Q$   $x$  and  $R$   $x$  be the quotient polynomial and remainder polynomial of the  $r$ -th recursion step with the initial condition that  $Q$   $x$  is 0 and  $R$   $x$  is a  $x$ .

So, you can write out these 2 recursive steps by dividing with respect to  $g$  of  $x$  and by forming this recursions you can always get the answer. So the circuit implementation of this division by  $g$  of  $x$  is written again like this again very very easy to implement in hardware.

(Refer Slide Time: 40:23)

Information Theory, Coding and Cryptography

## Example

The shift register circuit for dividing by  
 $g(x) = x^8 + x^6 + x^5 + x^3 + x + 1$

32

Indian Institute of Technology,  
DelhiRanjan Bose  
Department of Electrical Engineering

So, if you are suppose given the task to find out a shift register based implementation by dividing by this specific generator polynomial, then here is the circuit which will do division by  $g$  of  $x$ .

(Refer Slide Time: 40:44)

Information Theory, Coding and Cryptography

## Meggit Decoder

- The received word is first stored in a buffer.
- It is subjected to divide-by- $g(x)$  operation.
- The remainder in the shift register is then compared with all the possible (pre-computed) syndromes.
- These set of syndromes correspond to the set of correctable error patterns.
- If a syndrome match is found, the error is subtracted out from the received word.
- The corrected version of the received word is then passed on to the next stage of the receiver unit for further processing.

33

Indian Institute of Technology,  
DelhiRanjan Bose  
Department of Electrical Engineering

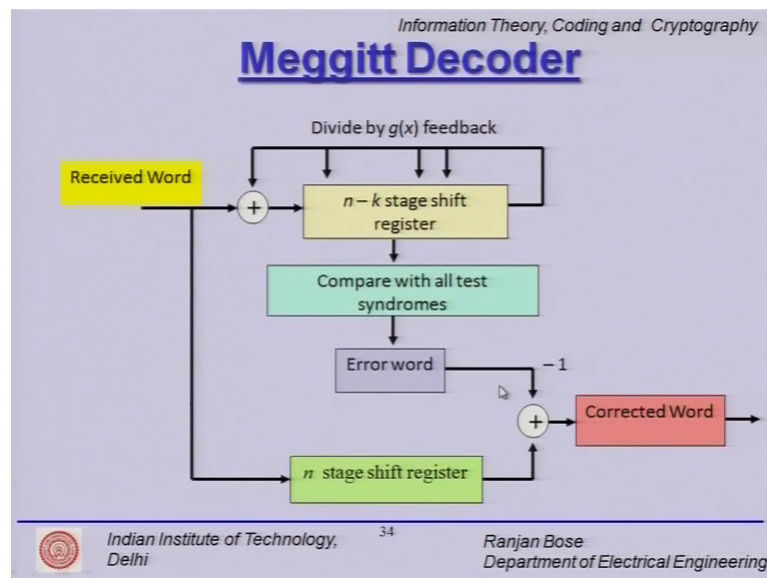
We finally, come to the flowchart of how we do the decoding using a Meggitt decoder for cyclic code. So first what you do is take the received word and store it in a buffer, buffer of what; it is nothing but shift register. Now we have a very efficient divide by  $g$  of  $x$  operation, so we subjected to the divide by  $g$  of  $x$  operation then the remainder in the



shift register is compared with all the possible pre computed syndromes what is remainder in the shift register is what to divide and get after division by  $x$ .

If it is all 0's then you have correctly decoded and it means that no error had happened, but if it is non zero then you have to compare, once again comparison in hardware is extremely easy so you have got very fast comparators available; so this is again very very easy and efficient to do it in hardware. Now once you compare an identifier syndrome it corresponds to a correctable error pattern, so we have a look up table again in hardware. So once the syndrome match it found the error is subtracted out from the received word all in hardware and then the corrected version of the received word is passed on to the next stage for processing.

(Refer Slide Time: 42:16)



So if you want to represent the Meggit decoder by a flowchart you have a received word first step is divide by  $g$  of  $x$  operation we have looked at some examples, you compare it with all the test syndromes and you get an error word hopefully it is in your list and what you do is take that error word here and then subtract out the error word and you get the corrected word. So this each and every step here is easily durable in hardware. Therefore, cyclic codes are so attractive because they are very easily implemented in hardware.

(Refer Slide Time: 42:58)

Information Theory, Coding and Cryptography

## Summary

- Generator Polynomial
- Fire Code, Golay Code
- CRC Codes
- Circuit Implementation
- Meggitt Decoder

Indian Institute of Technology, Delhi 35 Ranjan Bose  
Department of Electrical Engineering

With that we come to the end of this lecture. Let us summarise what we have done; so we started off with the generator polynomial and with that we also found out the syndrome polynomial. Then we looked at some examples of fire code very strong burst error correcting code, we looked at the perfect code the Golay binary code, the Golay ternary code, then we looked at cyclic redundancy check CRC codes very efficient in detecting errors, burst errors. And then finally, we looked at how to implement the cyclic codes using circuits, we also look at the flowchart of the Meggitt decoder.

With that we come to an end of this lecture.