

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

**Module - 26**  
**Convolutional Codes**  
**Lecture - 26**

Hello and welcome to our next module on Convolutional Codes.

(Refer Slide Time: 00:33)

Information Theory, Coding and Cryptography

## Outline

- Tree Codes and Convolutional Codes
- Trellis Codes
- Encoding using Trellis
- Polynomial Description
- Generator Polynomial Matrix

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

Let us start with a brief outline for today's talk. We would discuss what we mean by tree codes and convolutional codes followed by the notion of trellis codes. This is an interesting class of codes with memory then we will understand how we do encoding using trellis codes. And finally, we look at the polynomial description followed by the generator polynomial matrix specifically for convolutional codes. So, that is the brief outline for today's talk.

(Refer Slide Time: 01:08)

Information Theory, Coding and Cryptography

## Recap

- Linear Block Codes
- Cyclic Codes
- BCH Codes
- Reed-Solomon Codes

Indian Institute of Technology, Delhi

3

Ranjan Bose  
Department of Electrical Engineering

Now, let us see what we have done so far. So far we have looked into different kinds of linear block codes including cyclic codes, BCH codes, Reed Solomon codes, and several other examples, but there were all codes without memory. What we want to do today is look at a different class of codes with memory.

(Refer Slide Time: 01:32)

Information Theory, Coding and Cryptography

## Introduction

- So far we have studied **block codes**, where a block of  $k$  information symbols are encoded into a block of  $n$  coded symbols.
- There is always a **one-to-one correspondence** between the uncoded block of symbols (information word) and the coded block of symbols (codeword).
- This method is particularly useful for **high data rate applications**, where the incoming stream of uncoded data is first broken into blocks, encoded, and then transmitted
- Many of the good codes that have **large distance properties** are of large blocklengths (e.g., the RS codes),
- Larger **blocklengths** imply that the encoding overhead is small.

Indian Institute of Technology, Delhi

Ranjan Bose  
Department of Electrical Engineering

So, the block codes that we have studied so far always had a one to one correspondence between what is the input information block, or the information word and the output

word or the codeword always. So, if you have the same information word coming in you would always look up in the lookup table and give out the same codeword right.

So, this is useful for high data rate applications and the incoming stream of data is broken up into blocks, each block is encoded and we have seen very good codes like Reed Solomon codes with beautiful distance properties and very efficient code rates. Now the large block lengths also suffer from a basic disadvantage. Even though they give you very efficient codes we cannot declare the results of decoded output until the entire block is received and decoded.


(Refer Slide Time: 02:20)

*Information Theory, Coding and Cryptography*

## Large Blocklengths

- **Very large blocklengths** have the disadvantage that unless the entire block of encoded data is received at the receiver, the decoding procedure cannot start, which may result in delays.
- In contrast, there is another coding scheme in which much smaller blocks of uncoded data of length  $k_0$  are used.
- These are called **Information frames**.
- An information frame typically contains just a few symbols, and can have as few as just one symbol!
- These information frames are encoded into **Codeword Frames** of length  $n_0$ .
- However, just one information frame is not used to obtain the codeword frame.
- Instead, the current information frame with previous  $m$  information frames are used to obtain a single codeword frame.

---

 Indian Institute of Technology,  
DelhiRanjan Bose  
Department of Electrical Engineering

So, we cannot start declaring the results, so if the block consists of say 1024 bits and we are working at 10 kilobits per second. Then there is a tangible amount of delay possible before we start declaring results and this may not be very conducive for real time operations. But there is another class of codes that we are going to study today where the delay is minimal and that what we will study is the convolutional codes today which works using information frames of length  $k$  naught which are in just a few bits.

So, please note earlier the  $k$  the information word used to be 253 bits or a 473 bits, they large or even thousands of bits long. But here the information frames that we will deal with will be just a few bits 1, 2, 3. So, we will be working with much smaller sets of symbols which will be used to encode and each information frame would result in a codeword frame of length  $n$  naught.

So, what we are doing different is  $k$  naught symbols or bits will be encoded into  $n$  naught symbols or bits where both  $k$  naught and  $n$  naught are small. So, we would have not just the usage of the input frame information frame to generate the codeword frame we would also use memory ok. So, we would use the stored information of previous  $m$  information frames, and we would together use the current frame and the previous  $m$  frames to declare what is the resultant codeword frame.

So, it should be obvious that the same information frame may not lead to the same codeword frame because we do not know what is sitting in the memory.

(Refer Slide Time: 04:43)

*Information Theory, Coding and Cryptography*

### Codes with Memory

- Instead, the current information frame with previous  $m$  information frames are used to obtain a single codeword frame.
- This implies that such encoders have **memory**, which retain the previous  $m$  incoming information frames.
- The codes that are obtained in this fashion are called tree codes.
- An important sub-class of **Tree Codes**, used frequently in practice, is called **Convolutional Codes**.
- Up to now, all the decoding techniques discussed are algebraic and are memoryless *i.e.* decoding decisions are based only on the current codeword.
- Convolutional codes make decisions based on past information *i.e.* memory is required.

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

So, let us talk about codes with memory right. We have mentioned that previous  $m$  information frames are used to obtain a single codeword frame, so we are going to rely on memory right. So, such codes which are based on this memory and obtained in this fashion are called tree codes, a subclass of tree codes which we will use in real world are the convolutional codes we will formally define that today right.

Now, up to now the decoding techniques that we have discussed for linear block codes are algebraic and memoryless. But here we have to go one step further and figure out how to decode codes with memory because it is not just a lookup table we have to keep track of what was in the memory. So, the state the memory becomes important so, we must have a state space diagram to understand how the encoder is working. So, the



convolutional codes make decisions based on the past information and therefore, memory is an integral part.

If you look at a block encoder here you have this information frames coming in of size  $k$  and  $n$  blocks coming out, but we will have a different structure for codes with memory.


(Refer Slide Time: 06:18)

*Information Theory, Coding and Cryptography*

## Convolutional Codes

- We assume that we have an infinitely long stream of incoming symbols (thanks to the volumes of information to be sent these days, its not a bad assumption!).
- This stream of symbols is first broken up into segments of  $k_0$  symbols.
- Each segment is called an information frame, as mentioned earlier.
- The encoder consists of two parts :
  - memory, which basically is a shift register, and
  - a logic circuit.
- The memory of the encoder can store  $m$  information frames.
- Each time a new information frame arrives, it is shifted into the shift register and the oldest information frame is discarded.
- At the end of any frame time the encoder has  $m$  most recent information frames in its memory, which corresponds to a total of  $mk_0$  information symbols.

---

 Indian Institute of Technology,  
DelhiRanjan Bose  
Department of Electrical Engineering

So, what we assume is that we have an infinitely long stream of incoming symbols and this is not a bad assumption. Because we have an over deluge of data in today's world and we do not really complain about too many input symbols coming in.

So, first we break up this input bit stream into  $k$  naught symbol blocks or frames we will use the word frames to distinguish from the block codes that we have used earlier. Now each segment which is called a frame will be encoded, but the encoder is it consists of two parts a memory which is basically a shift register and a logic circuit associated with it. So, these two are the ingredients that form the encoder for a convolutional code.

The memory which is a shift register is capable of storing  $m$  previous information frames. So, each time a new information frame arrives it is shifted into the shift register, but in order to do. So since, the size of the shift register is limited the old the oldest information frame has to be discarded only then it will make room for the new information frame to come in. So, at any time  $m$  into  $k$  naught symbols must sit in the shift register which is the memory.


(Refer Slide Time: 07:52)

*Information Theory, Coding and Cryptography*

## Convolutional Codes

- When a new frame arrives, the encoder computes the codeword frame using this new frame that has just arrived and the stored previous  $m$  frames.
- The computation of the codeword frame is done using the logic circuit.
- This codeword frame is then shifted out.
- The oldest information frame in the memory is then discarded and the most recent information frame is shifted into the memory.
- The encoder is now ready for the next incoming information frame.
- Thus for every information frame ( $k_0$  symbols) that come in, the encoder generates a codeword frame ( $n_0$  symbols).
- It should be observed that the same information frame may not generate the same codeword frame because the codeword frame also depends on the  $m$  previous information frames.

---

 *Indian Institute of Technology,  
Delhi*

*Ranjan Bose  
Department of Electrical Engineering*

So, when a new frame arrives the encoder computes the codeword frame using the new frame and the stored previous  $m$  frame. And what do we mean by computes? It uses some kind of a logic circuit to do so. And once the computation is done the codeword frame is shifted out for transmission the oldest information frame in the memory is discarded because the new frame must be shifted in. And once this operation is complete we are now waiting for the next frame to come in and continue the process. Thus, for every  $k$  naught symbols that come in  $n$  naught symbols go out.

So, clearly the clock rates at the input and output are different please note convolutional encoders are very hardware fair friendly right. And since memory plays an integral part integral role in this, therefore the same  $k$  naught symbols are the input may, or may not generate the same  $n$  naught symbols at the output.


(Refer Slide Time: 09:00)

*Information Theory, Coding and Cryptography*

### Constraint length

- The **Constraint Length** of a shift register encoder is defined as the number of symbols it can store in its memory.
- We shall give a more formal definition of constraint length later in this chapter.
- If the shift register encoder stores  $m$  previous information frames of length  $k_0$ , the constraint length of this encoder  $v = mk_0$ .

---

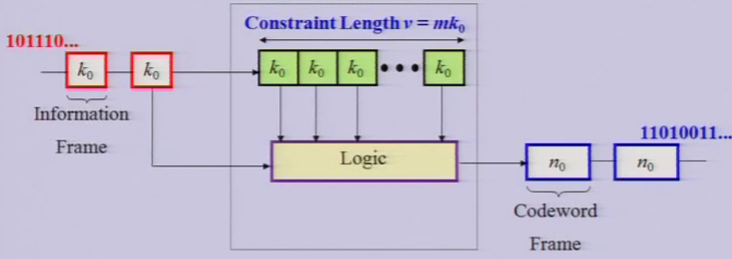
 Indian Institute of Technology, DelhiRanjan Bose  
Department of Electrical Engineering

So, let us now define something called as a constraint length of a shift register encoder. So, the constraint length is defined as the number of symbols it can store in its memory in terms of the input frames that is coming in right.

(Refer Slide Time: 09:20)


*Information Theory, Coding and Cryptography*

### Encoder for a Tree Code



A shift register encoder that generates a tree code.

---

 Indian Institute of Technology, DelhiRanjan Bose  
Department of Electrical Engineering

So, if you look at a general structure for a tree code it can be expressed as follows. So, let us start on the left hand side, we have the information frames coming in you have this long stream of input data say 101110 so and so forth. We first break it up into

information frames of size  $k$  naught, my  $k$  naught can be as small as 1 it could be 2, 3 4, but whatever it is we break the input into  $k$  naught frames.

Now, if you see this dotted block is the encoder it has two components the memory which is the shift register and the logic is pretty much a set of x-or operations. Now if we look at the shift register here you will note that there are a certain number of information frames that can be stored in this shift register. And let us say  $n_u$  equal to  $m$  times  $k$  naught, where  $k$  naught is the size of the information frame then the constraint length  $n_u$  equal to  $m$  times  $k$  naught.

So, it tells me how many previous information frames have been stored in the memory? An intuitive understanding tells us that the larger the constraint length the stronger my code can be because I am not relying on more of the past information. Now once we have this going then the logic can be used to compute the codeword frame  $n$  naught, as you can see the codeword frame is longer than the information frame. If information frame is say 2 bit long then the codeword frame could be 3, 4, 5, if the information frame is 3 bits long, it could be 4 5 or 6.

(Refer Slide Time: 11:41)

*Information Theory, Coding and Cryptography*


### Tree Codes

- The infinite set of all infinitely long codewords obtained by feeding every possible input sequence to a shift register encoder is called an  $(n_0, k_0)$  **Tree Code**.
- The rate of this tree code is defined as

$$R = \frac{k_0}{n_0}$$

- A more **formal definition**: A  $(n_0, k_0)$  **tree code** is a mapping from the set of semi infinite sequences of elements of  $GF(q)$  into itself such that if for any  $M$ , two semi infinite sequences agree in the first  $Mk_0$  components, then their images agree in the first  $Mk_0$  components.

---

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

So, this structure generates for every  $k$  naught symbols or bits coming in  $n$  naught codeword frames. So, this infinitely long code words because input is infinite, so output is infinite it keeps going right it is called an  $n$  naught comma  $k$  naught tree code it is

distinguished from n comma k block code. So, we use n naught comma k naught where n naught is the output frame k naught is the information frame.

So, codeword frame n naught information frame k naught, and the code rate will clearly be k naught over n naught and you can have a more formal definition in terms of the image. So, a tree code is a mapping from a set of semi infinite sequences of the elements of GF q because we are not necessarily limiting us to binary into itself such that if for any M, two semi infinite sequences agree in the first M k naught component, then the image will also agree in the first M k naught components.

(Refer Slide Time: 12:45)


*Information Theory, Coding and Cryptography*

### Definitions

- The **Wordlength** of a shift register encoder is defined as  $k = (m + 1)k_0$ .
- The **Blocklength** of a shift register encoder is defined as
 
$$n = (m + 1)n_0 = k \frac{n_0}{k_0}$$
- Note that the **code rate**

$$R = \frac{k_0}{n_0} = \frac{k}{n}$$


---

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

So, let us look at certain other definitions which will be useful to understand the tree codes first the word length. The word length can be defined in terms of this m remember m was used to define the constraint length. So, m plus one into k naught is k, which we defined as the word length and the block length n can be m plus 1 into n naught.

So, I can have a parallel understanding of the word length and the block length for the a tree code and code rate is nothing but k naught over n naught which can be easily expressed as your n and k the ratio of the word length to the block length.


(Refer Slide Time: 13:33)

*Information Theory, Coding and Cryptography*

### More definitions

- Normally, for practical shift register encoders, the information frame length  $k_0$  is small (usually less than 5).
- Therefore, it is difficult to obtain the code rate  $R$  of tree codes close to unity, as is possible with block codes (e.g., RS codes).
- **Definition** An  $(n_0, k_0)$  tree code that is linear, time-invariant, and has a finite wordlength  $k = (m + 1)k_0$  is called an  $(n, k)$  **Convolutional Code**.
- **Definition** An  $(n_0, k_0)$  tree code that is time-invariant and has a finite wordlength  $k$  is called an  $(n, k)$  **Sliding Block Code**.
- **Thus, a linear sliding block code is a convolutional code.**

---

 Indian Institute of Technology,  
DelhiRanjan Bose  
Department of Electrical Engineering

So, let us have some more definitions, so for practical shift register encoders to work the information stream length is typically small, there is a major difference from a linear block codes where  $k$  is huge. So, it could be 5 or less 7's are also common, but we would rather have small  $k$  naught.

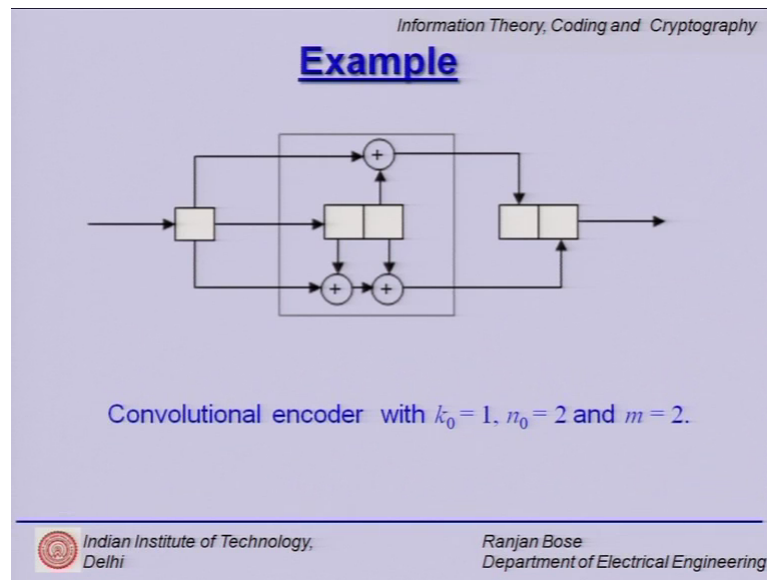
So, code rate of this tree codes are not really close to unity because if  $k$  naught is for example, 5 then  $n$  naught can be 6, leading us to the best possible code rate of 5 over 6 we could have 5 over 7 5 over 8, but then we are looking it worse and worse. So, let us now understand what do we mean by a convolutional code which is actually used in real world application?

So, a  $n$  naught comma  $k$  naught tree code that is linear and time invariant and has a finite word length  $k$  given by this is called an  $n$  comma  $k$  convolutional code. So, we can easily see that the convolutional code is a subclass of a tree code,  $n$  naught comma  $k$  naught tree code that is time invariant and has a finite word length  $k$  is also called as a sliding block code. Therefore, a linear sliding block code is a convolutional code.

So, please note we are still in the dome domain of linear codes ok. So, linearity means the all 0 codeword would necessarily be a part of it and sum of two code words would also be a valid codeword in the definition of code words that we have just defined.



(Refer Slide Time: 15:24)



Now, let us understand this interesting concept using a simple example. So, here if you see on the left hand side your  $k$  naught is 1, you shift register has 2 bits and your output  $n$  naught has 2 bits. So, it is clearly a rate 1 by 2 encoder with  $k$  naught equal to 1,  $n$  naught is equal to 2, and  $m$  equal to 2 and the logic is as follows whatever is in this memory binary addition with the current input binary addition with the second stored bit gives you the output bit 2, an output bit 1 is nothing but the present incoming bit xored with the second stored bit will give you this output bit.

So, it is very quickly you one can calculate the output based on the input it goes as fast as the hardware circuit. And we know that a typical delay today in current circuit is less than a nanosecond, so we can have really fast encoders.


(Refer Slide Time: 16:47)

*Information Theory, Coding and Cryptography*

### Example

- This encoder takes in one bit at a time and encodes it into 2 bits.
- The information frame length  $k_0 = 1$ , the codeword frame length  $n_0 = 2$  and the **blocklength**  $(m + 1)n_0 = 6$ .
- The **constraint length** of this encoder  $\nu = 2$ .
- The **code rate** of this encoder is  $\frac{1}{2}$ .
- The clock rate of the outgoing data is **twice** as fast as that of the incoming data.
- The adders are binary adders, and from the point of view of circuit implementation, are simply XOR gates.

---

 Indian Institute of Technology,  
DelhiRanjan Bose  
Department of Electrical Engineering

So, this example takes one bit at a time and codes it into 2 bits leading at to a code rate of 1 by 2. The information frame length is  $k_0$  equal to 1, so for the block length is  $m$  plus 1 times  $n_0$   $m$  is 2, 3,  $m$  plus 1 is 3 times  $n_0$  is 2 is 6. So, the effective block length is 6, constraint length of the same code rate is  $\nu$  equal to 2, code rate half.

As we can say we have already observed and a clock rate of the outgoing data must be twice the input clock rate simply because for every one bit that is pumped into that encoder two bits must go out. So, the adders are all binary adders, so binary adders without carry which is simply the XOR gates ok, so very very hardware friendly implementation.


(Refer Slide Time: 17:45)

*Information Theory, Coding and Cryptography*

### Example

- Let us assume that the initial state of the shift register is [0 0].
- Now, either '0' will come or '1' will come as the incoming bit.
- Suppose '0' comes.
- On performing the logic operations, we see that the computed value of the codeword frame is [0 0].
- The 0 will be pushed into the memory (shift register) and the rightmost '0' will be dropped.
- The state of the shift register remains [0 0].
- Next, let '1' arrive at the encoder.
- Again we perform the logic operations to compute the codeword frame.
- This time we obtain [1 1].
- So, this will be pushed out as the encoded frame.
- The incoming '1' will be shifted into the memory, and the rightmost bit will be dropped.
- So the new state of the shift register will be [1 0].

---

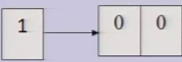
 *Indian Institute of Technology,  
Delhi* *Ranjan Bose  
Department of Electrical Engineering*

So, how does this encoder work in this example? So, basically we assume that the initial shift register at state is 00 that is the initial condition.

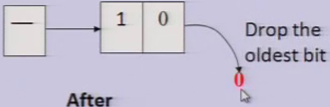
(Refer Slide Time: 18:00)

*Information Theory, Coding and Cryptography*

### Example




**Before**



**After**

- Again, there are two possibilities regarding the incoming bit: '0' or '1'.
- It is possible to construct a table which lists all the possibilities.

---

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

Now input will be either a 1 or a 0 and so what happens is if the initial condition is 00, for the shift register. We are only focusing on the shift register and the input rest all we have blanked out suppose the input is 1, then this 0 will go here, this 1 will go here, and the older 0 must be thrown out. So, after shifting in this portion 1 has shifted here, the 0 here

has shifted here, and the previous 0 has been thrown out discarded. So, we dropped the oldest bit and we can do.


(Refer Slide Time: 18:36)

*Information Theory, Coding and Cryptography*

### Example

The incoming and outgoing bits of the convolutional encoder

Incoming Bit	Current State of the encoder		Outgoing Bits	
0	0	0	0	0
1	0	0	1	1
0	0	1	1	1
1	0	1	0	0
0	1	0	0	1
1	1	0	1	0
0	1	1	1	0
1	1	1	0	1

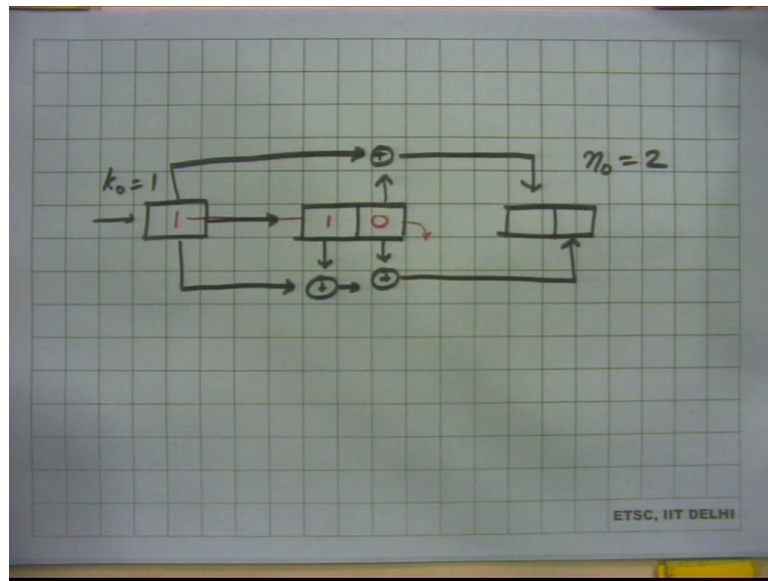


*Indian Institute of Technology,  
Delhi*

*Ranjan Bose  
Department of Electrical Engineering*

So, for all possible combinations of input bit, and the current state of the encoder, so, we did that example for 00, what if the 0 bit comes in 00. What if 1 bit input bit 1 comes in if the state of the encoder is 011011. So, clearly there are 8 possible combinations, so there are 8 entries into the table. And for each one of the input bit and the correct current state we have a calculation of the output bit based on the original state of the encoder which was as follows. So, whatever is the input bit and we have an output bit going.

(Refer Slide Time: 19:30)

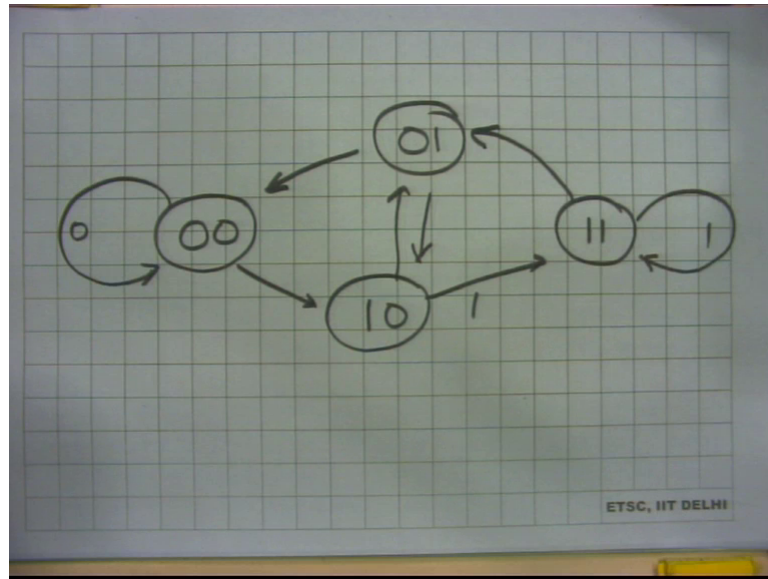


So, let us look at this example a little bit more carefully. So, we have an input block and then we have 2 bits as the encoder, and we have this going in here, binary adder here and this goes to your output. Similarly you have a binary adder here, the input bit comes in here and you have 1 here so  $n_0 = 2$  and  $k_0 = 1$ .

Now if you were to look at the table you will realize that whatever is sitting here. If it was a 00 here, and a 1 comes in here, then this 1 shifts here and the older 0 shifts here, and the last one is discarded. And once you have that then these calculations can be conducted and you can do this basic calculation and it will give you the desired output right.

So, if we go back to our slides and we will see that it is possible to generate a table of input bits, current state of the encoder and the output bits ok. Now suppose if you look at the next slide we can represent this as a simple state diagram. So, we can have the state diagram of the encoder in the slides, if you can see the state diagram you have a 00 as one of the states, 01 as one of the other states, 10 and 11.

(Refer Slide Time: 21:25)

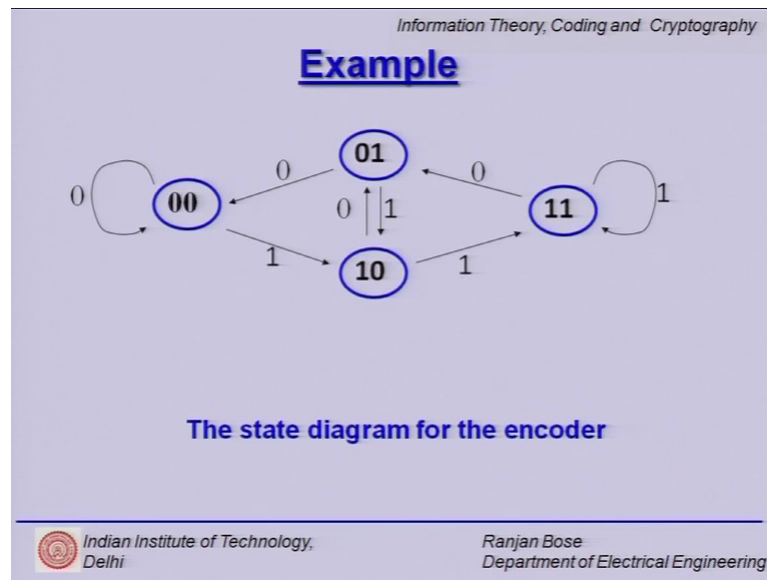


So, we have we have only 4 states, and we can see that depending upon whichever state you are sitting in if a 0 comes in your state does not change. But if a 1 comes in your state changes, if you are sitting in state 10 and if a 0 comes you become 01, but if a 1 comes you become 11 because the older bit will be thrown out this bit will be shifted here and a 1 bit will be shifted in the first place.

If you are sitting in 11, and a 1 comes you will retain your location as 11, but; however, if a 0 comes in the older bit will be thrown out the 0 will move in here, and the older 1 will move to this place leading it to a 01. Similarly if you are at 01, and a 1 comes then you will go to 10. So, you can make this a simple state diagram that you do for typical circuits.



(Refer Slide Time: 22:36)



So, we go to our slide where we show that we have the state diagram for the encoder this particular encoder. And these state diagram interconnections will change, if I change the logic circuit.


(Refer Slide Time: 22:50)

Information Theory, Coding and Cryptography

### Example

- We observe that there are **only  $2^2 = 4$  possible states** of the shift register.
- The bits associated with each arrow represent the incoming bit.
- It can be seen that the same incoming bit gets encoded *differently* depending on the current state of the encoder.
- This is different from the linear block codes studied in previous chapters where there is always a one-to-one correspondence between the incoming uncoded block of symbols (information word) and the coded block of symbols (codeword).

---

 Indian Institute of Technology, DelhiRanjan Bose  
Department of Electrical Engineering

So, only 4 possible states are there and we have already represents the bits associated with each arrow in terms of the incoming bits. So, it can be seen that the same input bit gets encoded differently depending on the current state of the encoder, and in this way

this code with memory is different from the linear block code which gives the same output for the same input.


(Refer Slide Time: 23:19)

*Information Theory, Coding and Cryptography*

## Trellis Diagram

- The same information contained in the state diagram can be conveyed usefully in terms of a graph called the **Trellis Diagram**.
- A trellis is a graph whose nodes are in a rectangular grid, which is semi infinite to the right.
- Hence, these codes are also called **Trellis Codes**.
- The number of nodes in each column is finite.

---

 Indian Institute of Technology, Delhi

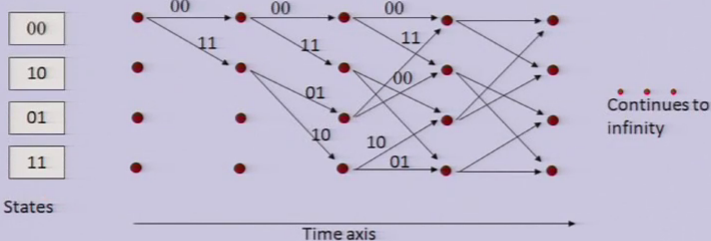
*Ranjan Bose  
Department of Electrical Engineering*

Now if you look at the state diagram the same information can be represented in a much more friendly manner, using a Trellis diagram. So, what is a trellis? A trellis is a graph whose nodes are arranged in a rectangular grid which is semi infinite to the right, and therefore, this code is also called as a trellis code. The number of nodes in each column is finite and is limited to the number of states.

(Refer Slide Time: 23:48)

*Information Theory, Coding and Cryptography*

## Trellis Diagram




Time axis

Continues to infinity

**The trellis diagram corresponding to the encoder**

---

 Indian Institute of Technology, Delhi

*Ranjan Bose  
Department of Electrical Engineering*

So, let us look at the same example the same state diagram now being recast as the trellis stack. On the left hand side we have the different states as you know there are only 4 states, 00, 10, 01, and 11. So, each of these nodes represent a state on the x axis we have the time axis that is each time an input symbol comes in a state transition may or may not happen, but we will move to the next level and the next level and the next level.

So, initially we are at state 00 represented by level 1, now if a 0 comes in you can see from your state diagram that you will retain your location in the state diagram as 00, so you continue to be the top node is the 00 state, but if a 1 appears as a input think a state transition will happen and you will move to 10.

So, there is an arrow connecting here, but please note regardless of whether a 0 comes in or a one comes in you can never go from state 00 to state 01 or from 00 to 11. So, certain state transitions are just not possible the transmitter knows this is the receiver knows this, but a noise does not know. So, this geometric structure together with the algebraic structure will add to the strength of our trellis codes ok, this is the point to be observed that we have now 2 r assistance a geometric structure as well.

Student: Sir.

Yes.

Student: So, but we can design the logic circuit instruction will get it may jump from 00 to 11.

So the question being asked is can we design our circuit diagram to have a transition from 00 to 11 sure you can, but please note then some other transition will not be possible. Because there will be only two branches emanating from every node every node is a state, one branch would correspond to input being 0, other should be corresponding to input being 1, there are only two possible inputs 0 or 1.

Student: But for a one we can for 0 it is going to (Refer Time: 26:17) with 00, but for the one incoming bit it may go to 01 or 11 that (Refer Time: 26:22).

Yes it can be possible that we can have a connection between this node and this node, but then there will be no connection between this node and this node. The point is no node is holier than the other node for me these are just 4 states, 4 nodes, whether it is moving

from node 1 to node 2 versus node 1 to node 3. But it will not be able to transit from one node 1 to node 2 and 4.

So, some transitions are not legal that is the point of this exercise I know it receiver knows that noise does not. So, I will use this geometric structure also to help me reconstruct what was sent and it will be used the same trellis diagram will be used to decode the transmitted bit stream it is an important observation. So, if you continue with this trellis diagram notion then you will realize that for whatever state you are in there will be two emanating branches 1 for 0 input, and 1 for a 1 input.

On the top of each branches is the output of the encoder written. So, since it is  $n$  naught equal to 2, you will always find two bits written on every branch. So, this is a recast of the state diagram in terms of a trellis diagram, see it is semi infinite we know where it starts it does not end until infinity. So, it continues to infinity. Therefore, we call it semi infinite.


(Refer Slide Time: 28:08)

*Information Theory, Coding and Cryptography*

## Trellis Diagram

- Every node in the trellis diagram represents a **state** of the shift register.
- Since the rate of the encoder is  $\frac{1}{2}$ , one bit at a time is processed by the encoder.
- The **incoming bit** can be either a '0' or a '1', therefore, there are two branches emanating from each node.
- The **top branch** represents the input as '0' and the lower branch corresponds to '1'.
- The output of the encoder is written on top of that branch.

---

 Indian Institute of Technology,  
Delhi

Ranjan Bose  
Department of Electrical Engineering

So, we have noted that every node is a state and then the rate is 1 by 2. So, 1 bit comes in and 2 bits go out, so, input bit can be a 1, or a 0 corresponding to two emanating branches from each node we can say that the top branch represents the input 0, and the bottom one represents a input as 1. And so output of the encoder is written on top.


(Refer Slide Time: 28:34)

*Information Theory, Coding and Cryptography*

## Trellis Diagram

- A trellis diagram gives a very easy method to encode a stream of input data.
- **Encoding procedure:**
- We start from the top left node (since the initial state of the encoder is  $[0\ 0]$ ).
- Depending on whether a '0' or a '1' comes, we follow the upper or the lower branch to the next node.
- The encoder output is read out from the top of the branch being traversed.
- Again, depending on whether a '0' or a '1' comes, we follow the upper or the lower branch from the current node (state).
- Thus, the encoding procedure is simply following the branches on the diagram and reading out the encoder outputs that are written on top of each branch.

---

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

So, this trellis diagram can be used to encode the input bit stream very very easily because we already have built in the time axis in there. It is not so using the state diagram where we have the same information, but not represented in along the time axis. So, encoding is very simple encoding is just following directions along the trellis diagram. So, we start with the top left node now if a 0 comes in follow the upper branch, 1 comes in follow the lower branch right.

So in fact, the input bit streams is 0111001 it is telling you go up up down down up up up down along the trellis. So, if you look at this trellis diagram, again if you have a 0, a 0 comes in we move here, 1 comes in we will go down, 1 comes in we move down, 0 comes in we go up up down down up and we just read out whatever is written on top of that branch that the transition along and that is the output. So, the input stream is just a direction that whenever you hit a node whether you go on top or you go on bottom.

So, it is extremely easy to encode it also shows a very important thing any particular bit stream to be encoded follows a path in the trellis which is a unique. So, input bit stream is corresponding to a unique path in the trellis there is no other way out. So, the decoding problem should be finding out the most likely path in the trellis and that is what we will pursue.


(Refer Slide Time: 30:30)

*Information Theory, Coding and Cryptography*

### Example

- Suppose we have to encode the bit stream **1 0 0 1 1 0 1 ...**
- We start from the top left node and follow the branches dictated by this input bit stream, *i.e.*, **down, up, up, down, down, up, down ...**, because '0' implies take the upper branch (up) and '1' implies take the lower branch (down).
- The encoded path in the trellis diagram is shown in Fig. 6.6.
- The encoded sequence can be read out from the diagram as **11 01 11 11 10 10 00 ...**.
- It can be seen that there is a one-to-one correspondence between the encoded sequence and a path in the trellis diagram.

---

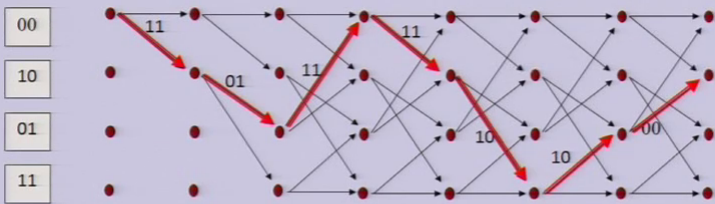
 Indian Institute of Technology, DelhiRanjan Bose  
Department of Electrical Engineering

So, let us say we have to encode this random bit stream 1001101 dot dot dot dot dot. So, we start with the top left and the moment we hit a 1, we take the lower branch 0, hit the upper branch hit the upper branch lower lower upper lower. So, it is up down up down and we can now look at the trellis diagram here.

(Refer Slide Time: 30:54)

*Information Theory, Coding and Cryptography*


### Encoding using a Trellis



States

Encoding the input sequence 1 0 0 1 1 0 1 using the trellis diagram.  
The encoded sequence can be read off from the diagram as 11 01 11 11 10 10 00

---

 Indian Institute of Technology, DelhiRanjan Bose  
Department of Electrical Engineering

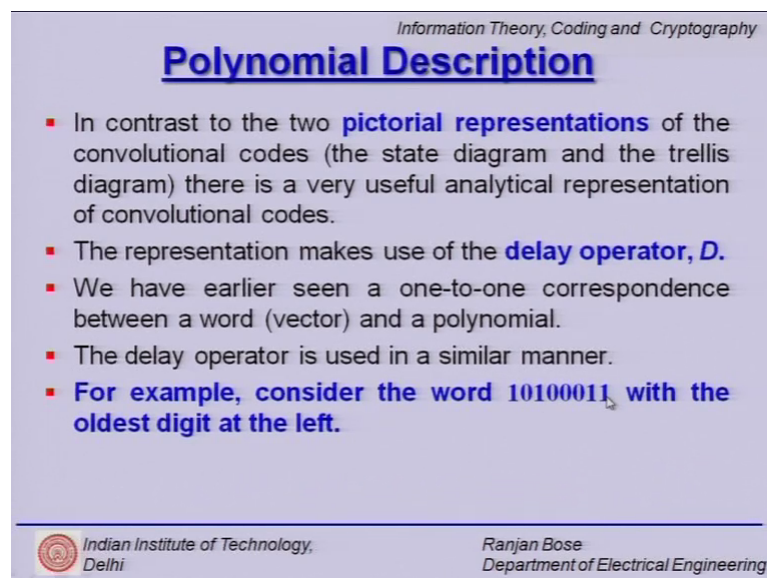
So, since the input was 1001101 1 means down, 0 means well we have two options go up take the upper branch take the upper route. Another 0 means of these two branches which are emanating out please take the upper branch. Now I have reached here and I look at



the input reaches it says 1 take the lower branch, and it says 1 take the lower branch. Because please note along the x axis is the time axis as the input bit stream comes in I follow the paths.

So, I am sitting here and now I again look at 0, I take the upper branch I take get 1, I read the lower branch. And now I just read out what is written on top because the encoder for that state and input is already pre-calculated. So, I read out 11 01 11 11 10 10 00 and that is my nice looking output bit stream. For 7 bits of input we have 14 bits of output it is not a surprise because the rate is 1 by 2. So, clearly any particular input bit stream is a unique path in the trellis that I have shown here the red path is 10 01 011 input bit stream.

(Refer Slide Time: 32:30)



The slide is titled "Polynomial Description" and is part of a presentation on "Information Theory, Coding and Cryptography". It contains a bulleted list of points:

- In contrast to the two **pictorial representations** of the convolutional codes (the state diagram and the trellis diagram) there is a very useful analytical representation of convolutional codes.
- The representation makes use of the **delay operator,  $D$** .
- We have earlier seen a one-to-one correspondence between a word (vector) and a polynomial.
- The delay operator is used in a similar manner.
- **For example, consider the word 10100011 with the oldest digit at the left.**

At the bottom of the slide, there is a logo for the Indian Institute of Technology, Delhi, and the name of the speaker, Ranjan Bose, Department of Electrical Engineering.

Now, we have a very nice succinct polynomial description. So, so far we have considered pictorial representation in terms of state diagram and trellis diagram we will now go for a polynomial description using a delay operator. Why a delay operator because it is very handy we have already a shift register where every shifting operation is a delay operation.

So, we already know that there is a one to one correspondence between a vector and a polynomial we have seen that in earlier convolutional codes. We will use the delay operator in a similar manner. So, we will have for example, this 10100011 with oldest

digit to the left and the most recent digit to the right as it enters the convolutional encoder.

(Refer Slide Time: 33:25)

*Information Theory, Coding and Cryptography*


### Polynomial Description

- The analytical representation (sometimes referred to as the *transform*) of this information word  $I(D)$  will be

$$I(D) = 1 + D^2 + D^6 + D^7.$$

- The **Indeterminate**  $D$  is the number of time units of delay of that digit relative to the chosen time origin, which is usually taken to coincide with the first bit.
- In general, the sequence  $i_0, i_1, i_2, i_3, \dots$  has the representation  $i_0 + i_1D + i_2D^2 + i_3D^3 + \dots$

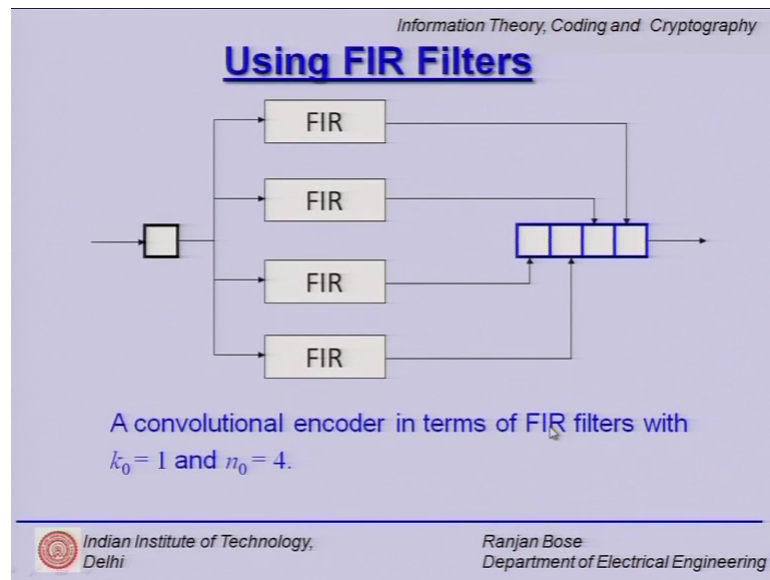
---

 Indian Institute of Technology, Delhi

*Ranjan Bose  
Department of Electrical Engineering*

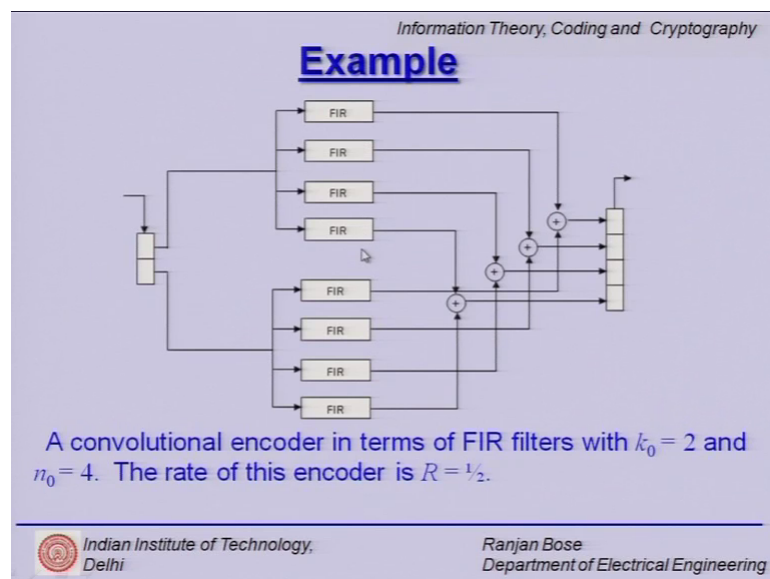
So, the analytical expression will be also sometimes referred to as the transform in terms of  $I(D)$  is the following. So, it is nothing but 10100011 the coefficients of  $D$ ,  $D$  cubed  $D$  raised power 4 except trellis all 0 they are corresponding to the 0's in the bit stream, the  $D$  is the indeterminate and is the delay unit. So, consequently any sequence  $i_0, i_1, i_2, i_3$  can be represented as a  $i_0$  plus  $i_1D$  indeterminate plus  $i_2D^2$  squared. So, it tells you where in the location in terms of time frame my bits are.

(Refer Slide Time: 34:13)



So, we can have very efficient FIR based representation. What do you have is an input bit stream and of which  $k$  naught is the symbol frame input frame we have FIR here and then linked to the output. These FIR's are nothing but memory and logic.

(Refer Slide Time: 34:35)



Similarly, I can have a more complex setup here in in example  $k$  naught is 2, as you can see  $n$  naught is 4. So, the rate is 1 by 2 and you can have additional FIR filters, so this is to illustrate that it is extremely hardware friendly this is a salt problem we can build very efficient encoders for convolutional codes.


(Refer Slide Time: 35:00)

Information Theory, Coding and Cryptography

### Example

- Convolutional filter with  $k_0 = 2$  and  $n_0 = 4$ .
- Each of the FIR filters can be represented by a polynomial of degree  $\leq m$ .
- The input stream of symbols can also be represented by a **polynomial**.
- The operation of the filter can then simply be represented as the multiplication of the two polynomials.
- Thus the encoder (and hence the code) can be represented by a *set of polynomials*.
- These polynomials are called the **Generator Polynomials** of the code.
- This set contains  $k_0 n_0$  polynomials.

---

 Indian Institute of Technology, DelhiRanjan Bose  
Department of Electrical Engineering

So, we just now saw an example of  $k$  naught is equal to 2, and  $n$  naught is equal to 4, but can we represent this encoder using a polynomial matrix or generator polynomial matrix. Well, the answer is yes all we have to do is learn how to do an input output relationship.

(Refer Slide Time: 35:23)


Information Theory, Coding and Cryptography

### Generator Polynomial Matrix

- The largest degree of a polynomial in this set of generator polynomials is  $m$ .
- Recall that a block code was represented by a *single* generator polynomial.
- Thus we can define a **Generator Polynomial Matrix** of size  $k_0 \times n_0$  for a convolutional code as follows.

$$G(D) = [g_j(D)].$$

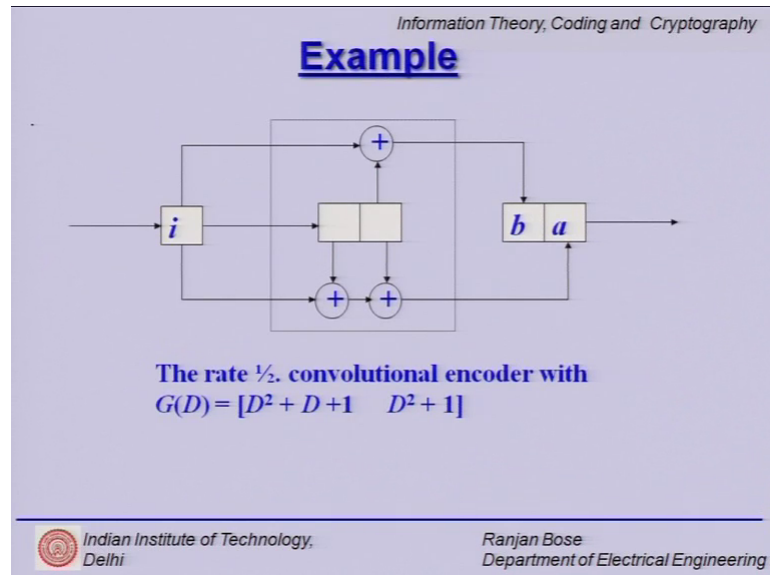
---

 Indian Institute of Technology, DelhiRanjan Bose  
Department of Electrical Engineering

So, the largest degree of a polynomial which in the set of generator polynomial is  $m$ , and a block code if you recall was used represented using a single generator polynomial, but here we will have a generator polynomial matrix linking the input to the output ok. So,

finally, we have to link say  $k$  naught input bits to  $n$  naught output bits through a delay register. So, let us look at a very simple example it is our previous example.

(Refer Slide Time: 36:00)



So,  $k$  naught is 1,  $n$  naught is 2, and then 2 elements in the shift register. But if you look at the output in this first bit well it takes the current bit the previous bit and previous to that bit. So, current no delay  $D$  raise power 0, 1 delay  $D$  raise power 1, 2 delays  $D$  raise power 2.

So, this first one is linked to the input as  $D^2 + D + 1$ , but if you look at  $b$  the second output we should be the second element of the matrix it is nothing but the input  $D$  raise power 0 plus not the first delay, but the previous to previous that. So,  $D^2$ , so  $D^2 + 1$ , so all I am doing is relating the input to the output through the series of delays right.

(Refer Slide Time: 37:00)

*Information Theory, Coding and Cryptography*


### Example

- The first bit of the output  $a = i_{n-2} + i_{n-1} + i_n$  and the second bit of the output  $b = i_{n-2} + i_n$ , where  $i_{n-l}$  represents the input that arrived  $l$  time units earlier.
- Let the input stream of symbols be represented by a polynomial.
- We know that multiplying any polynomial by  $D$  corresponds to a single cyclic right-shift of the elements.  
$$g_{11}(D) = D^2 + D + 1 \text{ and } g_{12}(D) = D^2 + 1,$$
- The **generator polynomial matrix** of this encoder can be written as

$$G(D) = [D^2 + D + 1 \quad D^2 + 1].$$

↘

---

 Indian Institute of Technology,  
Delhi

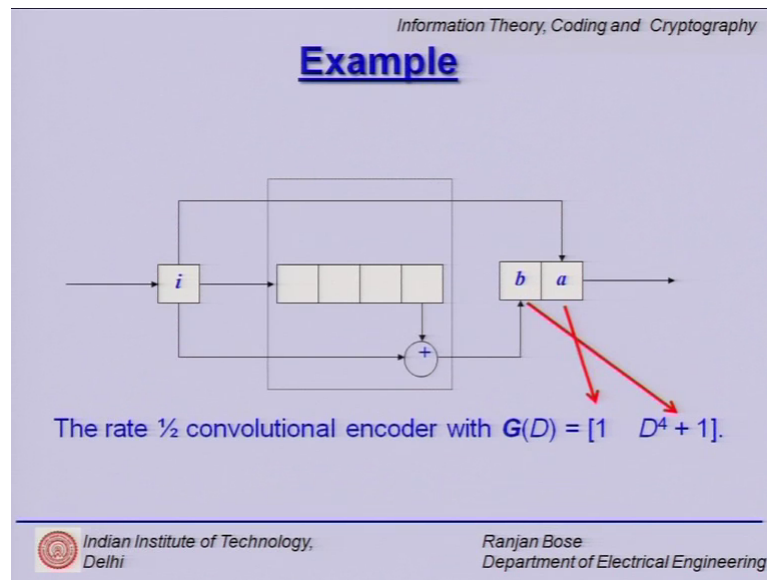
Ranjan Bose  
Department of Electrical Engineering

So, this is the example how we input and output are related and  $a$  and  $b$  are related using the  $D$  square. So, we have these two elements of the generator polynomial matrix  $g_{11}$  and  $g_{12}$  linking input one to output one and input one to output two. And so we have consequently the notion of a generator polynomial matrix which we will now write as  $D^2 + D + 1$  and  $D^2 + 1$ .

So, the same the entire hardware encoder can be represented using this generator polynomial matrix. So, if you give me the encoder in hardware I will give you the generator polynomial matrix, if you give me the generator polynomial matrix I will give you the hardware implementation of the encoder ok.



(Refer Slide Time: 38:00)



So, let us look at another simple example again we have 4 memory elements, well I first look at a, a this a is nothing but the input nothing else. So, it gives me a feeling that it should be just 1, and it is going to be a systematic convolutional encoder because the first bit goes right like that.

The second is their parity and how are we calculating the parity well it is the same bit and XORed with 4 bits that came before in the bit stream. So, it is delay D 4 plus 1 ok, so this is the generator polynomial matrix for this, so a is 1, and b is D 4 plus 1.

(Refer Slide Time: 38:49)

*Information Theory, Coding and Cryptography*

### Example

- **Generator polynomial matrix** of this encoder can be written as  
$$G(D) = [1 \quad D^4 + 1].$$
- Note that the first  $k_0$  bits ( $k_0 = 1$ ) of the codeword frame is identical to the information frame.
- Hence, this is a **Systematic Convolutional Encoder**.

Indian Institute of Technology, DelhiRanjan Bose  
Department of Electrical Engineering

So, we have the generator polynomial matrix representation as follows and we can say that this is clearly a systematic convolution encoder.

(Refer Slide Time: 39:00)

*Information Theory, Coding and Cryptography*

### Example

- Consider the systematic convolutional encoder represented by the following circuit

- The generator polynomial matrix of this encoder can be written as

$$G(D) = \begin{bmatrix} g_{11}(D) & g_{12}(D) & g_{13}(D) \\ g_{21}(D) & g_{22}(D) & g_{23}(D) \end{bmatrix} = \begin{bmatrix} 1 & 0 & D^3 + D + 1 \\ 0 & 1 & 0 \end{bmatrix}$$

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

So, if you look at this example again we have now input  $k_1$  and  $k_2$ , output  $n_1$ ,  $n_2$  and  $n_3$  again we have 3 bits and we have clearly a logic units. Please note in this example we have interestingly put  $k_1$  just going into this 1 ok. So, the constraint length will be defined with respect to only  $k_1$ , please note this 3 units in the memory is not a multiple of this  $k_1$ ,  $k_2$  and  $k$  naught equal to 2. So, now, we have 3 outputs related to 2 inputs, so there should be 6 elements in this, so it is a 2 cross 3 generator polynomial matrix.

Now, if you look at it  $n_1$  is nothing but  $k_1$ , so it is just a 1  $n_2$  is directly linked to  $k_2$  looks like we are again  $n$  for a systematic convolutional encoder. So, this is this  $g_{22}$  is 1 node,  $n_1$  is no way linked to  $k_2$ , so it is 0,  $n_2$  is no way linked to  $k_1$  it does no connection. So, it is a 0, so you have a small identity matrix right here and here the fun begins  $n_3$  is related to  $k_1$ . So,  $g_{13}$  is now 1 plus a delay plus  $D$  cubed this is  $D^3 + D + 1$ . So,  $n_3$  is  $D^3 + D + 1$ , but  $n_3$  in no way is related to  $k_2$ , so there is a 0. So, just by a visual inspection I can write the generator polynomial matrix.

So, we will use visual inspection.


(Refer Slide Time: 41:00)

*Information Theory, Coding and Cryptography*

## Writing the GPM

- It is easy to write the **generator polynomial matrix** by **visual inspection**.
- The element in the  $i^{\text{th}}$  row and  $j^{\text{th}}$  column of the matrix represents the relation between the  $i^{\text{th}}$  input bit and the  $j^{\text{th}}$  output bit.
- To write the generator polynomial for the  $(i^{\text{th}}, j^{\text{th}})$  entry of the matrix, just trace the route from the  $i^{\text{th}}$  input bit to the  $j^{\text{th}}$  output bit.
- If no path exists, the generator polynomial is the zero polynomial, as in the case of  $g_{12}(D)$ ,  $g_{21}(D)$  and  $g_{23}(D)$ .
- If only a direct path exists without any delay elements, the value of the generator polynomial is unity, as in  $g_{11}(D)$  and  $g_{22}(D)$ .
- If the route from the  $i^{\text{th}}$  input bit to the  $j^{\text{th}}$  output bit involves a series of memory elements (delay elements), represent each delay by an additional power of  $x$ , as in  $g_{13}(D)$ .

---

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

And immediately write the generator polynomial matrix there just trying to write the elements of  $i$ -th row and  $j$ -th column using the input output relationship. So, if no path exists then it is 0, as we see saw for  $g_{12}$ ,  $g_{21}$ , and  $g_{23}$  there is absolutely no connection between say  $k_2$  and  $n_3$ .

And if there is a direct path then it is unity as we saw in  $g_{11}$ , and  $g_{22}$  and thereby we got the identity matrix. And just all is you have to just observe how many delays you are going through.


(Refer Slide Time: 41:40)

*Information Theory, Coding and Cryptography*

## Summary

- Tree Codes and Convolutional Codes
- Trellis Codes
- Encoding using Trellis
- Polynomial Description
- Generator Polynomial Matrix

---

 *Indian Institute of Technology,  
Delhi* 40 *Ranjan Bose  
Department of Electrical Engineering*

So, with that we come to the end of this lecture let us summarize what we have learnt so far. We started off with the definition of tree codes and convolutional codes, and then we found a very very interesting pictographic representation of trellis codes using trellis diagrams.

We learnt how to use this trellis diagram to encode very efficiently an input bit stream we also made a very very interesting observation that any particular input bit stream corresponds to a unique path in the trellis. We then went on to define what do we mean by polynomial description of convolutional codes. And finally, we were able to represent any encoder in a hardware using a generator polynomial matrix.

With that we come to the end of this current lecture.