

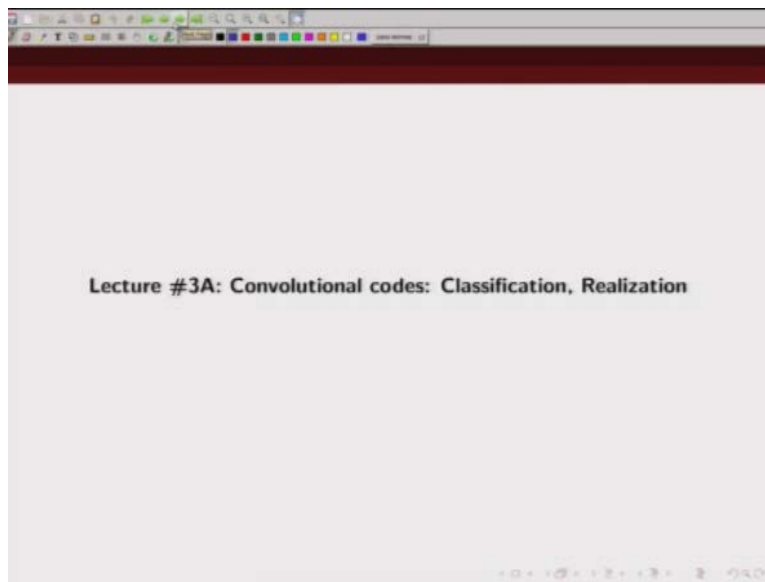
**Indian Institute of Technology Kanpur**  
**National Programme on Technology Enhanced Learning (NPTEL)**  
**Course Title**  
**Error Control Coding: An Introduction to Convolutional Codes**

**Lecture-3A**  
**Convolutional Codes: Classification, Realization**

by  
**Prof. Adrish Banerjee**  
**Dept. Electrical Engineering, IIT Kanpur**

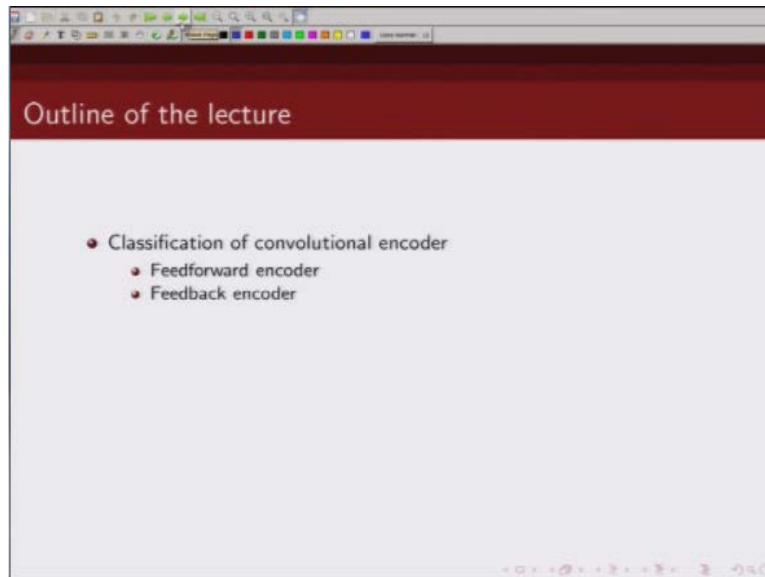
Welcome to the course on error control coding an introduction to convolutional codes. In this lecture.

(Refer Slide Time: 00:24)



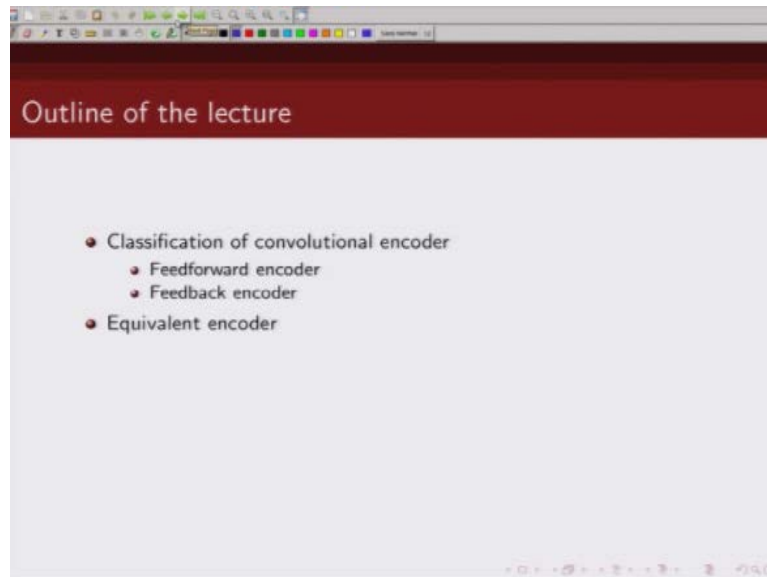
Today we are going to talk about classification of convolutional codes based on type of connections between the output and the input. Also based on what are our output bits, we will classify convolutional codes into systematic and nonsystematic codes. Then we are going to talk about how we can realize convolutional code using shift registers.

(Refer Slide Time: 00:55)



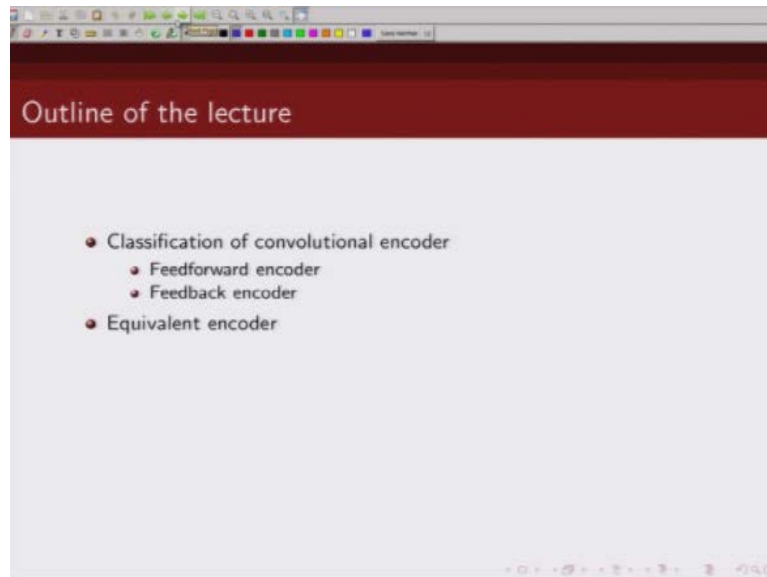
So at the set first we will talk about convolutional codes and in this we are going to talk about a classification based on types of connections between the input and output of the convolutional encoder. In this regard we are going to talk about what do we mean by feedforward encoder and feedback encoder.

(Refer Slide Time: 01:20)



Then we are going to introduce our classification based on what are the output bits whether the information bits directly appears in the output or not based on that, there will be a classification of convolutional code.

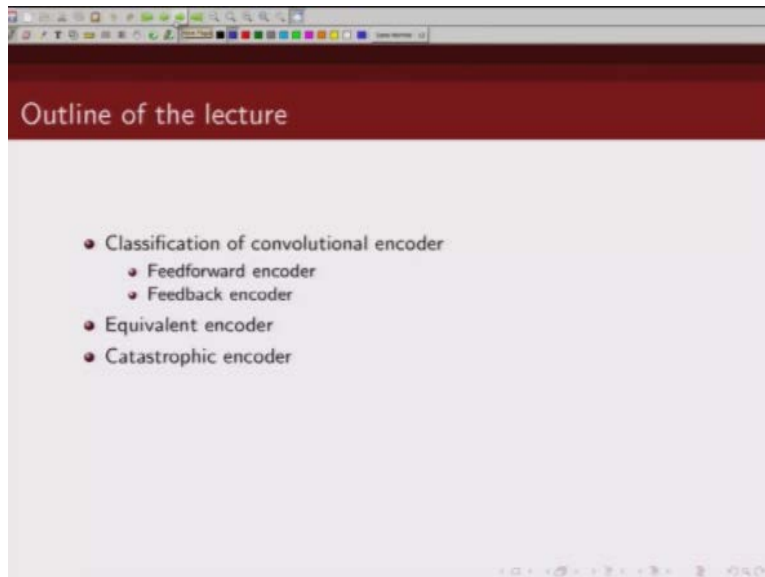
(Refer Slide Time: 01:39)



The encoder basically where information bits can be separated out is known as a systematic encoder and in a nonsystematic encoder we cannot separate out information bits directly from the parity bit. So we will talk about what we mean by a systematic encoder for a convolutional code and a nonsystematic encoder.

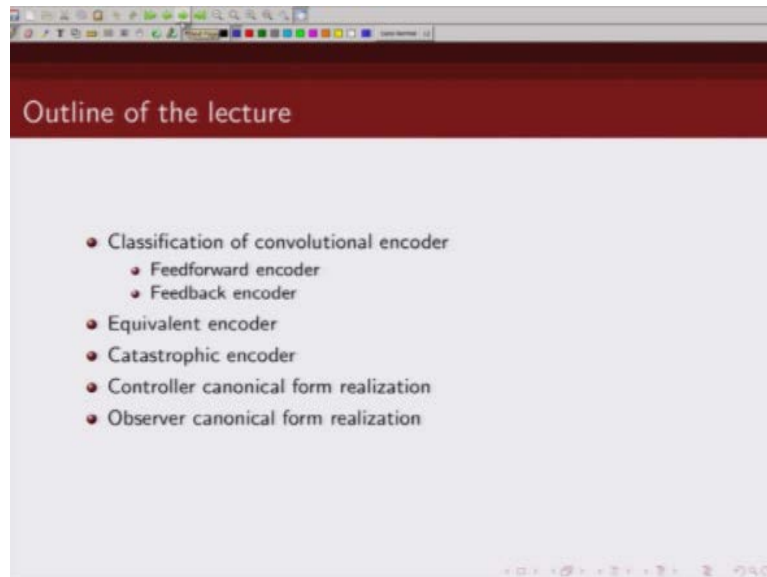
And then we will introduce this concept of equivalent encoders. So for every nonsystematic encoder there is an equivalent systematic and through an example we are going to illustrate how we can get its equivalent encoder.

(Refer Slide Time: 02:16)



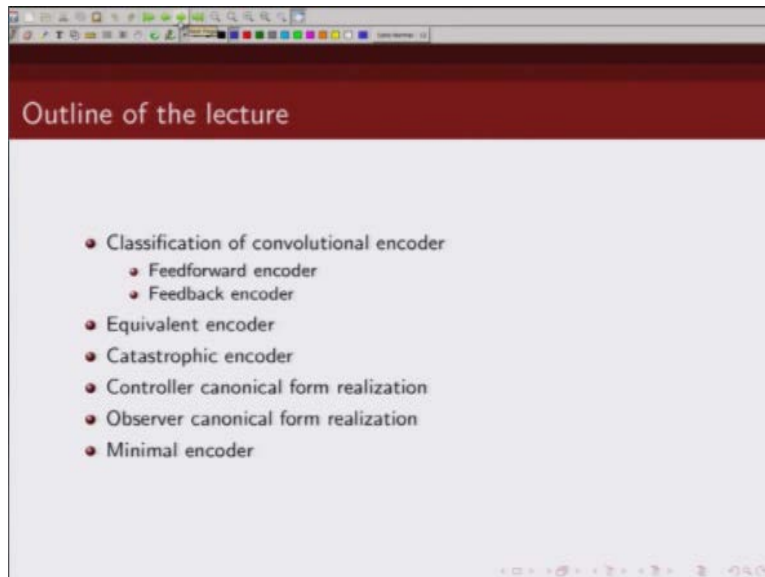
Then we are going to talk about a class of encoder where if the input bits are very high weight we can still get an output code word of very low weight and this kinds of encoders are known as catastrophic encoders.

(Refer Slide Time: 02:36)



And finally we are going to talk about two different types of realization of convolutional codes using shift register. The first one which is known as controller canonical form realization and the second one is known as observer canonical realization.

(Refer Slide Time: 02:55)



And finally we are going to conclude this lecture with the concept of minimal encoder.

(Refer Slide Time: 03:01)

The slide is titled "Classification of convolutional encoders" and discusses the "Feedforward Encoder". It includes a diagram of the encoder and its state transition logic.

**Feedforward Encoder:**

- The encoder corresponding to a polynomial generator matrix does not contain any feedback path, and hence it is known as a feedforward encoder.

**(a)** Block diagram of a feedforward encoder. An input  $u(D)$  enters a block, and the output  $v(D)$  is the sum of the input and a feedback path that branches off before the block and re-enters after it.

**(b)** State transition diagram with four states: 00, 01, 10, and 11. Transitions are labeled with input/output pairs: 0/0, 0/1, 1/0, and 1/1.

So let us start our discussion on classification of convolutional encoder the first type of encoder that we are going to talk about is known as feedforward encoder. So what is the feed forward encoder?



(Refer Slide Time: 03:15)

**Classification of convolutional encoders**

**Feedforward Encoder:**

- The encoder corresponding to a polynomial generator matrix does not contain any feedback path, and hence it is known as a feedforward encoder.

(a) Block diagram of a feedforward encoder. The input  $u(D)$  enters a block, and the output  $v(D)$  is produced. A feedback path is shown as a line that goes from the output back to the input, but it is not connected to the input, indicating no feedback.

(b) Generator matrix diagram. It shows a 2x2 grid of nodes. The top row nodes are labeled 0 and 0, and the bottom row nodes are labeled 1 and 1. The connections between nodes are labeled with polynomial terms: 0/0 (top-left to top-right), 0/1 (top-left to bottom-right), 1/1 (top-right to bottom-right), and 1/0 (top-right to bottom-left).

The encoder corresponding to a polynomial generator matrix which does not have any feedback from the output to the input is known as feedforward encoder.

(Refer Slide Time: 03:29)

**Classification of convolutional encoders**

**Feedforward Encoder:**

- The encoder corresponding to a polynomial generator matrix does not contain any feedback path, and hence it is known as a feedforward encoder.

(a) Block diagram of a feedforward encoder. The input  $u(D)$  enters a block. The output of the block is  $v(D)$ . A feedback path is shown from the output back to the input, but it is not connected to the block, indicating no feedback. The generator matrix is given as  $G(D) = 1 + D$ .

(b) State transition table for the encoder. The states are 00, 01, 10, and 11. The transitions are:

00	00	00
01	01	11
10	10	01
11	11	10

Let us take this example this is our information sequence  $v(D)$  denotes our coded sequence. What is the generator matrix  $G(D)$  in this case it is given by  $1+D$ . Note here the generator matrix here is a polynomial generator matrix right, as suppose to a rational generator matrix and there is no feedback from the output to the input side.

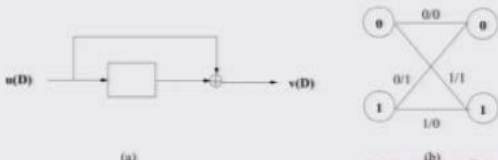
You can see basically the output depends on the current input as well as the input one pass time instance. So there is no feedback from the output to the encoder side. And this is an example of a feedforward encoder.

(Refer Slide Time: 04:27)

### Classification of convolutional encoders

Feedforward Encoder:

- The encoder corresponding to a polynomial generator matrix does not contain any feedback path, and hence it is known as a feedforward encoder.
- The output of a feedforward encoder can be represented as a linear combination of the current input and a finite number of past inputs. This is also referred as nonrecursive encoder.



(a) Block diagram of a feedforward encoder. The input  $u(D)$  enters a block, and the output  $v(D)$  is the sum of the current input and a delayed version of the input.

(b) Trellis diagram showing the state transitions. The states are labeled 0 and 1. The transitions are labeled with pairs of bits: 0/0, 0/1, 1/1, and 1/0.

Now we can represent the output of a feedforward encoder as linear combination of current input and finite number of past inputs. We also refer this type of encoder as nonrecursive encoder.

(Refer Slide Time: 04:47)

### Classification of convolutional encoders

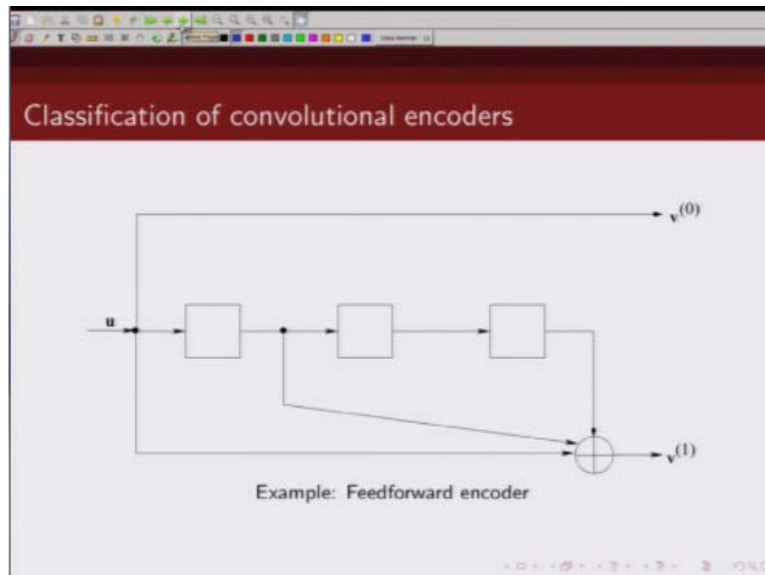
**Feedforward Encoder:**

- The encoder corresponding to a polynomial generator matrix does not contain any feedback path, and hence it is known as a feedforward encoder.
- The output of a feedforward encoder can be represented as a linear combination of the current input and a finite number of past inputs. This is also referred as nonrecursive encoder.
- In figure, the encoder diagram of a rate  $R = 1$ , 2-state feedforward encoder with generator matrix  $\mathbf{G}(D) = [1 + D]$  is shown using a shift register implementation.

(a) (b)

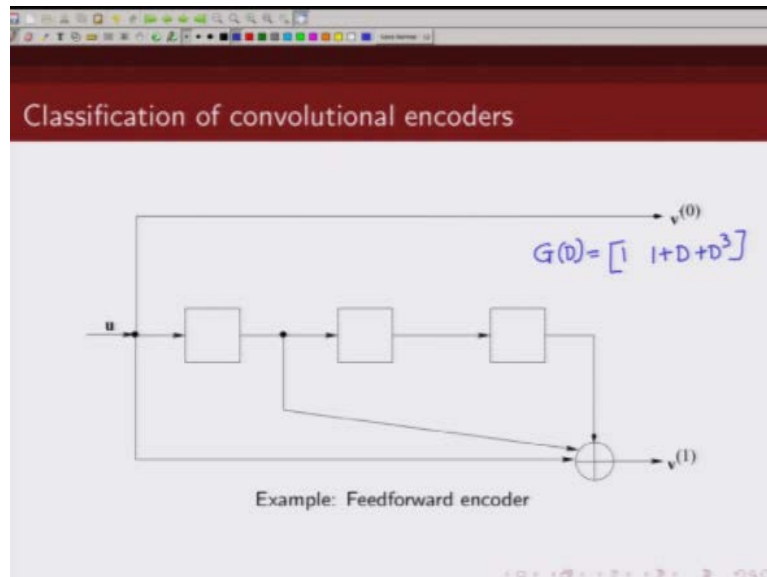
And as we said in this we have an example of a rate one code, because input to one bit, output there is one bit coming out, and the generator matrix of this rate one code is given by  $1+D$  and you can see it this is of example of a feedforward encoder whose generator matrix is a polynomial generator matrix and there is no feedback from the output to the input side. And this is this corresponding state diagram for this feedforward encoder.

(Refer Slide Time: 05:26)



This is another example of a feedforward encoder we can write down the generator matrix for this.

(Refer Slide Time: 05:35)



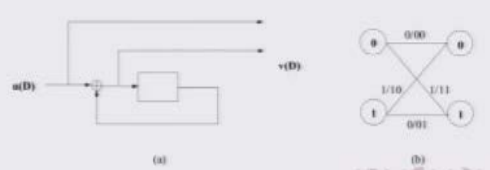
$G(D) v^{(0)}$  is nothing but the input bit so this first one is just 1. And what about the second parity bit this is information bit so we have 1 plus one delayed version of this information bit plus  $D^3$ . Because this is one, two, three, three time instance delayed version of  $u$ , so this is the generator matrix. This is also polynomial generator matrix and there is you can see, there is no feedback from the output to the input side.

(Refer Slide Time: 06:16)

Classification of convolutional encoders

Feedback Encoder:

- The encoder corresponding to a rational generator matrix with at least one nonpolynomial transfer function contains a feedback path and is known as a feedback encoder.



(a) Block diagram of a feedback encoder. The input  $u(D)$  enters a summing junction. The output of the summing junction goes to a block. The output of the block is  $v(D)$ . A feedback path branches off from the output  $v(D)$ , goes through a block, and is fed back into the summing junction.

(b) State transition diagram with four states: 00, 01, 10, and 11. Transitions are labeled with input/output pairs: 0/0, 0/1, 1/0, and 1/1.

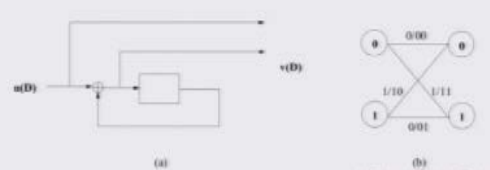
Now let us look at what do we mean by feedback encoder as opposed to a feedforward encoder.

(Refer Slide Time: 06:31)

### Classification of convolutional encoders

Feedback Encoder:

- The encoder corresponding to a rational generator matrix with at least one nonpolynomial transfer function contains a feedback path and is known as a feedback encoder.



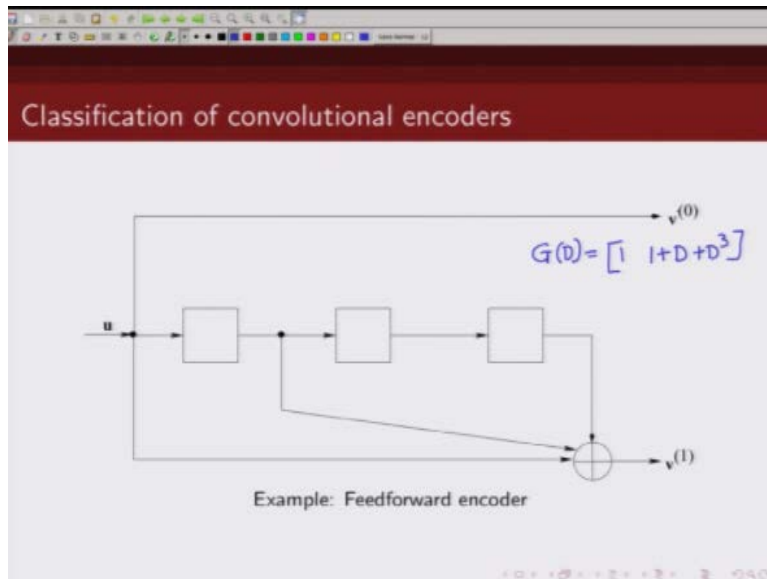
(a) Block diagram of a feedback encoder. The input  $u(D)$  enters a summing junction. The output of the summing junction is  $v(D)$ . A feedback path branches off from the output  $v(D)$ , passes through a delay element, and is subtracted from the input  $u(D)$  at the summing junction.

(b) Trellis diagram of the feedback encoder. The trellis has four nodes: 0 (top-left), 0 (top-right), 1 (bottom-left), and 1 (bottom-right). Transitions are labeled with branch labels: 0/00 (0 to 0), 1/10 (0 to 1), 0/01 (1 to 0), and 1/11 (1 to 1).

The encoder for a feedback encoder has a rational generator matrix please note here.



(Refer Slide Time: 06:36)



(Refer Slide Time: 06:36)

### Classification of convolutional encoders

Feedforward Encoder:

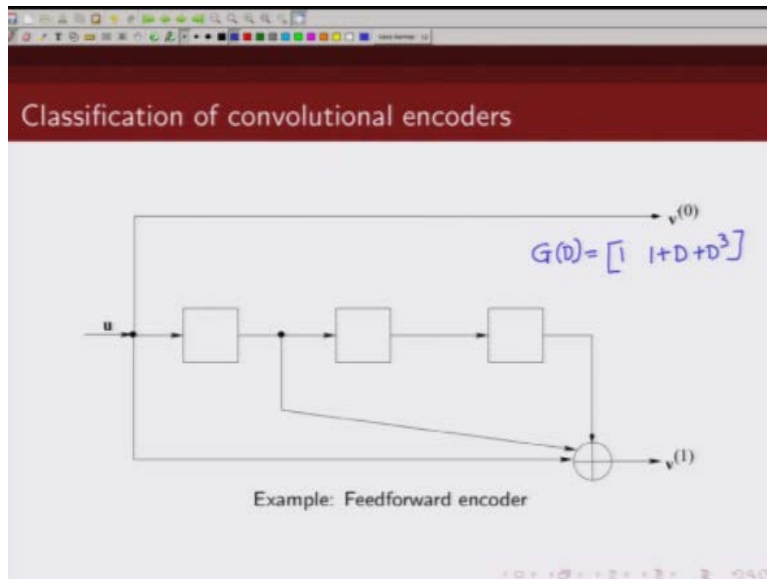
- The encoder corresponding to a polynomial generator matrix does not contain any feedback path, and hence it is known as a feedforward encoder.
- The output of a feedforward encoder can be represented as a linear combination of the current input and a finite number of past inputs. This is also referred as nonrecursive encoder.
- In figure, the encoder diagram of a rate  $R = 1$ , 2-state feedforward encoder with generator matrix  $\mathbf{G}(D) = [1 + D]$  is shown using a shift register implementation.

(a) Block diagram of a feedforward encoder. The input  $u(D)$  enters a block. The output of the block is  $v(D)$ . A feedback path is shown from the output back to the input, but it is a feedforward path, meaning it does not contain any feedback path.

(b) State transition diagram of a 2-state feedforward encoder. The states are labeled 0 and 1. The transitions are labeled with input/output pairs: 0/0, 0/1, 1/1, and 1/0.

We had a polynomial generator matrix for a feedforward encoder we had a polynomial generator matrix.

(Refer Slide Time: 06:47)



(Refer Slide Time: 06:47)

The slide is titled "Classification of convolutional encoders". Under the heading "Feedback Encoder:", there is a bullet point: "The encoder corresponding to a rational generator matrix with atleast one nonpolynomial transfer function contains a feedback path and is known as a feedback encoder." Below the text are two diagrams. Diagram (a) is a block diagram of a feedback encoder. It shows an input  $u(D)$  entering a summing junction. The output of the summing junction goes into a block representing a transfer function. The output of this block is  $v(D)$ . A feedback path branches off from the output  $v(D)$ , goes through a delay element (represented by a circle with 'D'), and then enters the summing junction. Diagram (b) is a state transition diagram with four states: 00, 01, 10, and 11. Transitions are labeled with input/output pairs: 00 to 00 is 0/00, 00 to 01 is 0/01, 01 to 00 is 1/10, 01 to 11 is 1/11, 10 to 01 is 0/00, 10 to 11 is 0/01, 11 to 10 is 1/10, and 11 to 00 is 1/00.

Whereas for a feedback encoder we have a rational generator matrix with at least one nonpolynomial transfer function containing a feedback path from the output to the input. Look at this example from the output we can see there is a feedback going to the input side.

(Refer Slide Time: 07:08)

## Classification of convolutional encoders

Feedback Encoder:

- The encoder corresponding to a rational generator matrix with atleast one nonpolynomial transfer function contains a feedback path and is known as a feedback encoder.

(a)

$$G(D) = \begin{bmatrix} 1 & 1/D \end{bmatrix}$$

(b)

And the generator matrix for this is basically – so first coded bit is nothing but the information bit so that is one, and this is basically  $1/1+D$ .

(Refer Slide Time: 07:25)

**Classification of convolutional encoders**

**Feedback Encoder:**

- The encoder corresponding to a rational generator matrix with atleast one nonpolynomial transfer function contains a feedback path and is known as a feedback encoder.
- The output of a feedback encoder can be represented as a linear combination of past inputs as well as past outputs.

(a) Block diagram of a feedback encoder. The input  $u(D)$  enters a summing junction. The output of the summing junction goes to a block representing the encoder. The output of the encoder is  $v(D)$ . A feedback path branches off from the output  $v(D)$ , goes through a delay element, and is fed back into the summing junction.

(b) Trellis diagram showing four states: 0, 1, 0, 1. Transitions are labeled with generator polynomials: 0/00, 1/10, 0/01, 1/11.

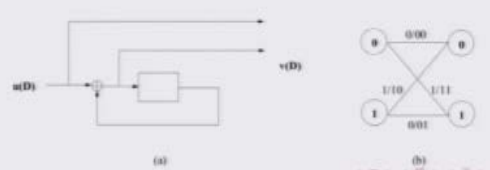
So, because there is a feedback from the output to the input side output of a feedback encoder can be written as a combination of past input as well as past outputs.

(Refer Slide Time: 07:42)

### Classification of convolutional encoders

Feedback Encoder:

- The encoder corresponding to a rational generator matrix with at least one nonpolynomial transfer function contains a feedback path and is known as a feedback encoder.
- The output of a feedback encoder can be represented as a linear combination of past inputs as well as past outputs.
- Hence the output depends on infinite number of past inputs. This is also sometimes referred as recursive encoder.



(a) Block diagram of a feedback encoder. The input  $u(D)$  enters a summing junction. The output of the summing junction goes to a block representing the encoder. The output of the encoder is  $v(D)$ . A feedback path branches off from the output  $v(D)$ , passes through a delay element, and is fed back into the summing junction.

(b) Trellis diagram showing the state transitions. The states are labeled 0, 1, 2, 3. The transitions are labeled with the input and output bits: 0/00, 1/10, 2/01, 3/11.

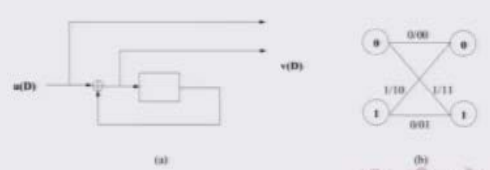
Hence the output depends on infinite number of past input.

(Refer Slide Time: 07:52)

### Classification of convolutional encoders

Feedback Encoder:

- The encoder corresponding to a rational generator matrix with atleast one nonpolynomial transfer function contains a feedback path and is known as a feedback encoder.
- The output of a feedback encoder can be represented as a linear combination of past inputs as well as past outputs.
- Hence the output depends on infinite number of past inputs. This is also sometimes referred as recursive encoder.



(a) Block diagram of a feedback encoder. The input  $u(D)$  enters a summing junction. The output of the summing junction goes to a block representing a transfer function. The output of this block is  $v(D)$ . A feedback path branches off from the output  $v(D)$ , goes through a delay element, and is added back to the input  $u(D)$  at the summing junction.

(b) State transition diagram. It shows four states: 00, 01, 10, and 11. Transitions are labeled with input/output pairs: 0/00, 0/01, 1/10, and 1/11.

Because the current output depends also on past output and past outputs also depends on past inputs and past output. So the output will basically depend on infinite number of past inputs. Now feedback encoder is also known as recursive encoder.

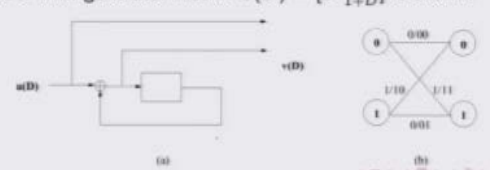


(Refer Slide Time: 08:16)

### Classification of convolutional encoders

**Feedback Encoder:**

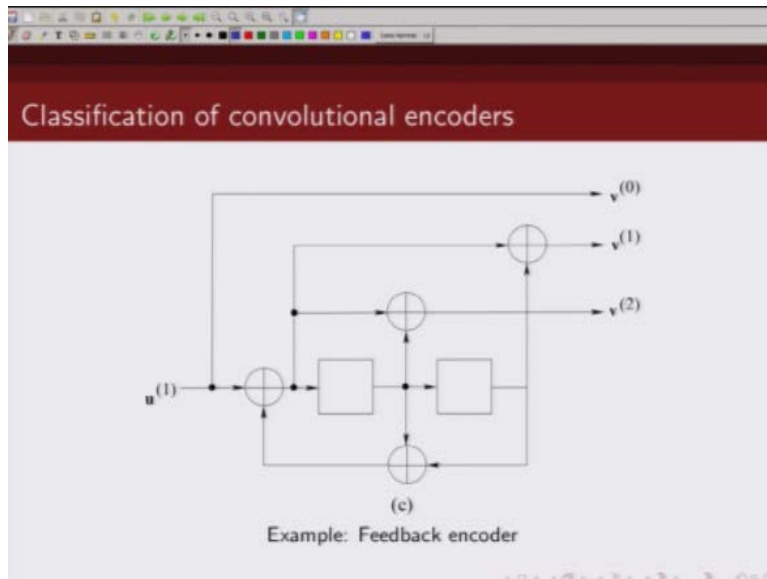
- The encoder corresponding to a rational generator matrix with atleast one nonpolynomial transfer function contains a feedback path and is known as a feedback encoder.
- The output of a feedback encoder can be represented as a linear combination of past inputs as well as past outputs.
- Hence the output depends on infinite number of past inputs. This is also sometimes referred as recursive encoder.
- In figure, the encoder diagram of a rate  $R = 1/2$ , 2-state feedback encoder with generator matrix  $\mathbf{G}(D) = \begin{bmatrix} 1 & \frac{1}{1+D} \end{bmatrix}$  is shown.



The figure consists of two parts: (a) an encoder diagram and (b) a state diagram. Part (a) shows a block diagram of a feedback encoder. An input  $u(D)$  enters a summing junction. The output of the summing junction is  $v(D)$ . A feedback path branches off from  $v(D)$ , passes through a delay element  $D$ , and is added back to the input at the summing junction. Part (b) is a state transition diagram with four states: 00, 01, 10, and 11. Transitions are labeled with input/output pairs: 00 to 00 (0/0), 00 to 01 (0/1), 01 to 00 (1/0), 01 to 11 (1/1), 10 to 01 (0/0), 10 to 11 (0/1), 11 to 10 (1/0), and 11 to 01 (1/1).

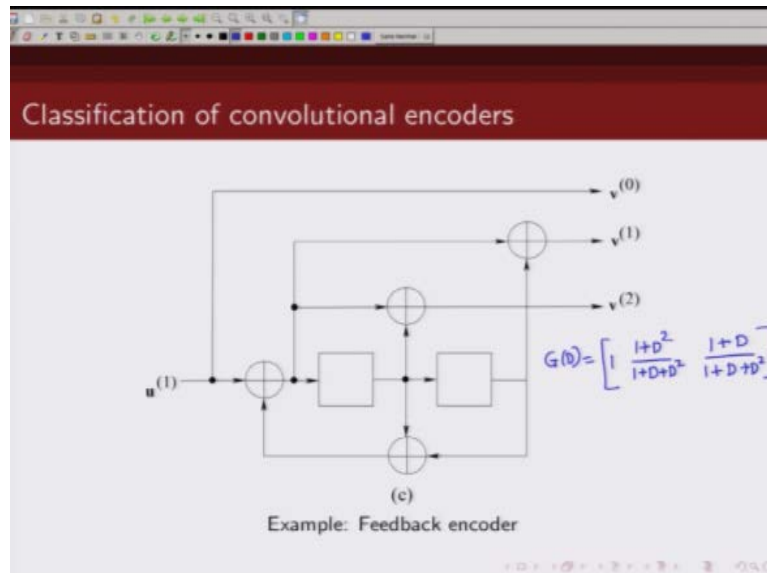
And we just now mentioned one example of this feedback encoder is given in this figure. This is a rate one half code you can see for one input, we have two outputs and its generator matrix is given by this. This is the corresponding state diagram for this feedback encoder.

(Refer Slide Time: 08:40)



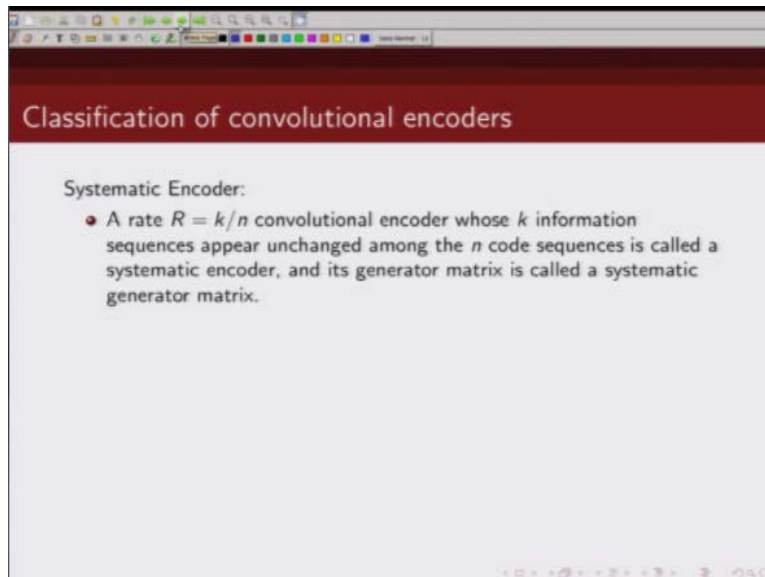
This is another example of a feedback encoder, so there is one input and there are three outputs.

(Refer Slide Time: 08:52)



We can write down the generator matrix  $G(D)$ , the first output is nothing but the information bits so that is one. Now what are the feedforward terms in  $v^{(1)}$  so  $D^1$  depends on this bit and this bit. So this is  $1+D^2$  and what is the denominating term, we have basically  $1+D+D^2$  term. Similarly  $v^2$  is basically given by  $1+D$  and this is  $1+D+D^2$ . So this is the generator matrix for this feedback encoder.

(Refer Slide Time: 09:44)



Classification of convolutional encoders

Systematic Encoder:

- A rate  $R = k/n$  convolutional encoder whose  $k$  information sequences appear unchanged among the  $n$  code sequences is called a systematic encoder, and its generator matrix is called a systematic generator matrix.

The next classification that we are going to talk about is based on output bits whether we can separate out the information bits from the coded bits.

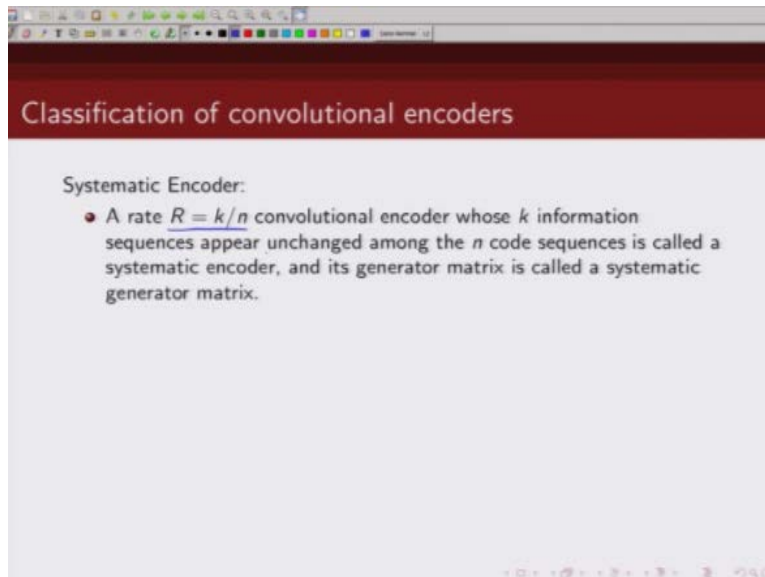
(Refer Slide Time: 09:59)

Classification of convolutional encoders

Systematic Encoder:

- A rate  $R = k/n$  convolutional encoder whose  $k$  information sequences appear unchanged among the  $n$  code sequences is called a systematic encoder, and its generator matrix is called a systematic generator matrix.

(Refer Slide Time: 10:02)



Classification of convolutional encoders

Systematic Encoder:

- A rate  $R = k/n$  convolutional encoder whose  $k$  information sequences appear unchanged among the  $n$  code sequences is called a systematic encoder, and its generator matrix is called a systematic generator matrix.

So in a systematic encoder a rate  $k/n$  systematic encoder, the  $k$  information bits appear unchanged in the output. So out of those encoded bits you can directly see the  $k$  information bits and rest  $n-k$ -bits are your parity bits.

(Refer Slide Time: 10:24)

### Classification of convolutional encoders

**Systematic Encoder:**

- A rate  $R = k/n$  convolutional encoder whose  $k$  information sequences appear unchanged among the  $n$  code sequences is called a systematic encoder, and its generator matrix is called a systematic generator matrix.
- In figure, a systematic rate  $R = 1/2$  feedback convolutional encoder is shown.

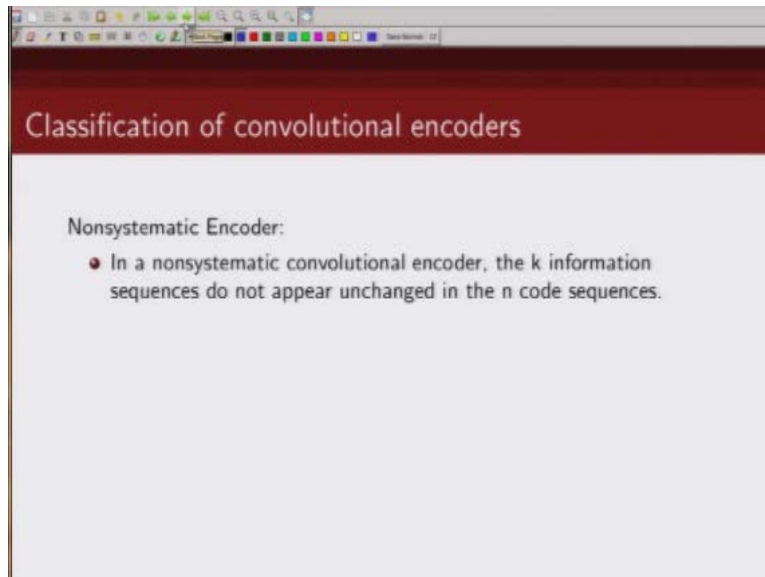
(a) Block diagram of a systematic rate  $R = 1/2$  feedback convolutional encoder. The input  $u(D)$  is fed into a feedback loop and a summing junction. The output  $v(D)$  is fed back into the feedback loop and also taken as the second output.

(b) Trellis diagram showing the state transitions for the encoder. The nodes are labeled 0 and 1. The transitions are labeled with generator polynomials: 0/00, 1/10, 0/01, and 1/11.

And the generator matrix corresponding to a systematic encoder is known as systematic generator matrix. Take example of this rate  $1/2$  feedback encoder you can see there is one input and there are two outputs so it is rate  $1/2$  and it is a feedback encoder there is a feedback from the output to the input side. You can see here the first coded bit is nothing but the information bit.

And the second coded bit is, is parity bit basically coming out from this convolutional encoder, so from these two coded bits we can easily find out what the information bit was from this bit so we can separate out the information bit from the coded bit and this is example of a systematic encoder.

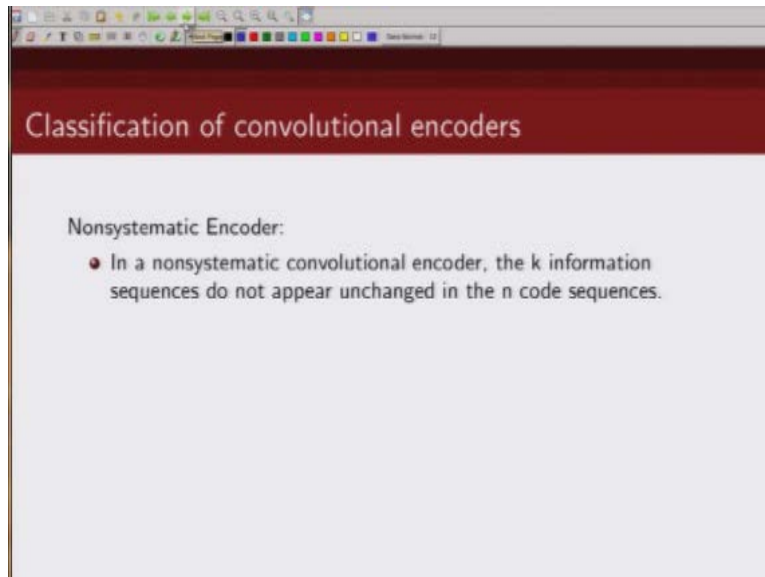
(Refer Slide Time: 11:31)



As opposed to a systematic encoder in a non systematic encoder.



(Refer Slide Time: 11:37)



The image shows a screenshot of a presentation slide. The slide has a dark red header with the title "Classification of convolutional encoders" in white text. Below the header, the text "Nonsystematic Encoder:" is followed by a bullet point: "• In a nonsystematic convolutional encoder, the  $k$  information sequences do not appear unchanged in the  $n$  code sequences." The slide is displayed in a window with a standard operating system taskbar at the top.

We cannot separate out the  $k$  information bits from the  $n$  coded bits.

(Refer Slide Time: 11:45)

### Classification of convolutional encoders

**Nonsystematic Encoder:**

- In a nonsystematic convolutional encoder, the  $k$  information sequences do not appear unchanged in the  $n$  code sequences.
- In figure, a nonsystematic rate  $R = 1/2$  feedforward convolutional encoder is shown.

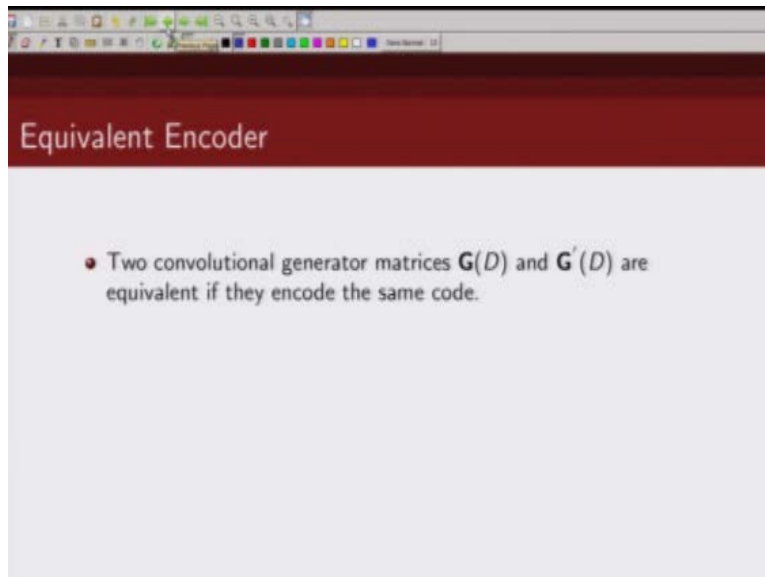
(a) (b)

This is an example of a one second I want to make it a rate 1 this is actually rate

(Refer Slide Time: 11:52)

Just type this is a rate 1 code because this 1 there is input and there is 1 output this is a rate 1 and this is a feedback output this is a rate 1 and this is a feedback feed forward encoder you can see there is no feedback from the output to the input side, so it is a rate 1 feed forward encoder and you can see the output bit is given by this current input bit and this past input bit so you cannot directly take out the information bits from this coded bit. So this is an example of a non systematic encoder.

(Refer Slide Time: 12:42)



The image shows a screenshot of a presentation slide. At the top, there is a dark red header bar with the text "Equivalent Encoder" in white. Below the header, the slide content is on a light gray background. A single bullet point is centered on the slide, stating: "Two convolutional generator matrices  $\mathbf{G}(D)$  and  $\mathbf{G}'(D)$  are equivalent if they encode the same code."

We could also define a class

(Refer Slide Time: 12:42)

### Classification of convolutional encoders

**Nonsystematic Encoder:**

- In a nonsystematic convolutional encoder, the  $k$  information sequences do not appear unchanged in the  $n$  code sequences.
- In figure, a nonsystematic rate  $R = 1/2$  feedforward convolutional encoder is shown.

(a)

(b)

Which is called partially systematic encoder so in a partially systematic so if you have rate  $k/n$  partially systematic encoder out of those  $k$  information bits some of them appear on change in the output while some of the information bits do not appear unchanged in the coded bits so in a systematic rate  $k/n$  encoder we can see directly the  $k$  information bits I a partial systematic encoder we can see a fraction of these  $k$  information bits may be few bits like from 1 to  $k-1$  and in an all systematic encoder we cannot see any systematic bits direct in any information bits directly in the output.

(Refer Slide Time: 13:45)

### Classification of convolutional encoders

**Nonsystematic Encoder:**

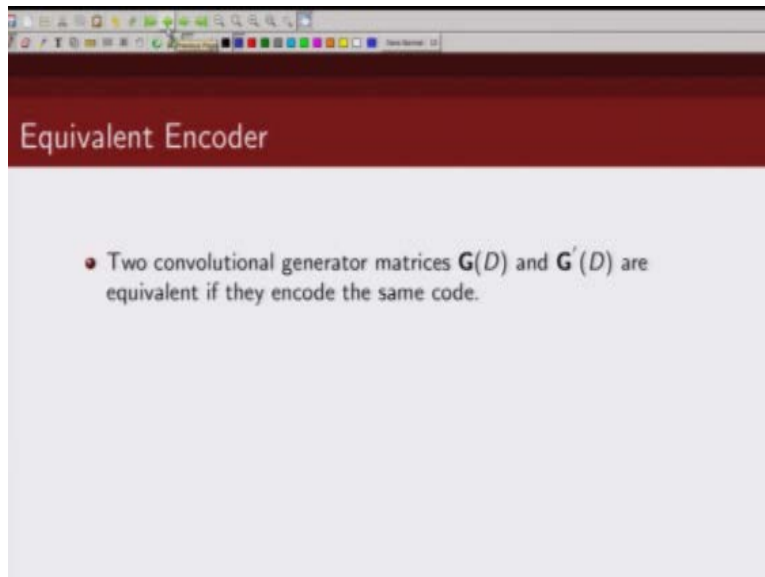
- In a nonsystematic convolutional encoder, the  $k$  information sequences do not appear unchanged in the  $n$  code sequences.
- In figure, a nonsystematic rate  $R = 1/2$  feedforward convolutional encoder is shown.

(a)

(b)

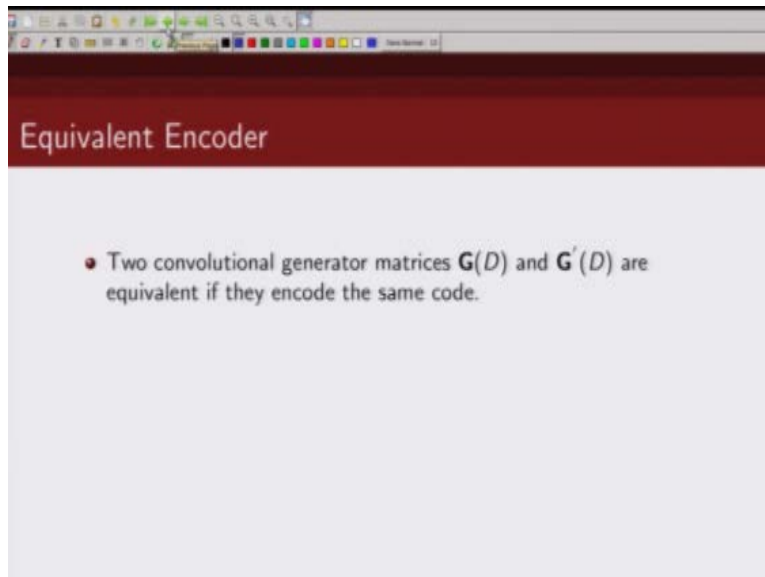
So all the parity bits essentially linear combination of current and pass inputs and outputs.

(Refer Slide Time: 13:45)



Now that brings us to our next topic of discussion which is a concept of equivalent encoders.

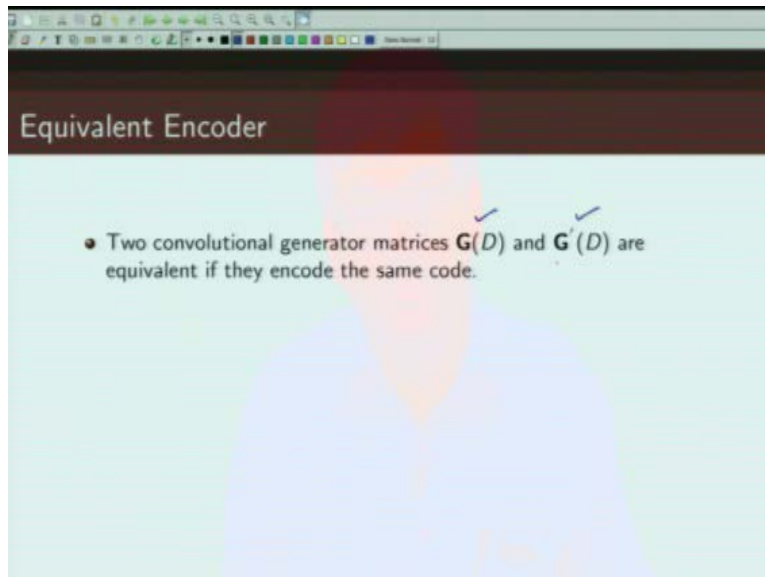
(Refer Slide Time: 14:04)



So before we defined what is an equivalent encoder we will defined what is an equivalent encoder we will defined what do we mean by equivalent generator matrices so we do convolutional generator matrix let us call it  $G(D)$  and  $G'(D)$  re equivalent if they encode the same code now what do we mean by encode the same code. So the set of

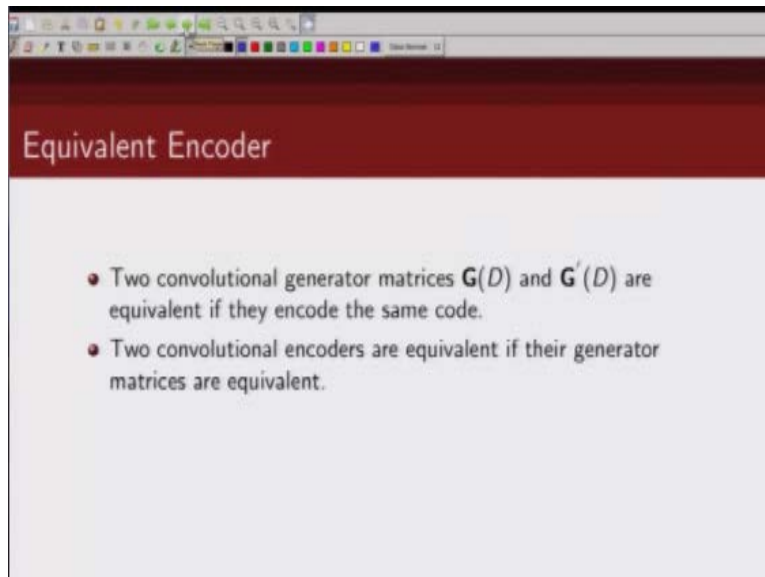


(Refer Slide Time: 14:32)



Code words generated by this and this if they are same then these generator matrices are equivalent now the set of code words generated by these generator matrix are same but the mapping between the input and the output is different in this encoder from what the mapping between inputs and output is for this generator matrix

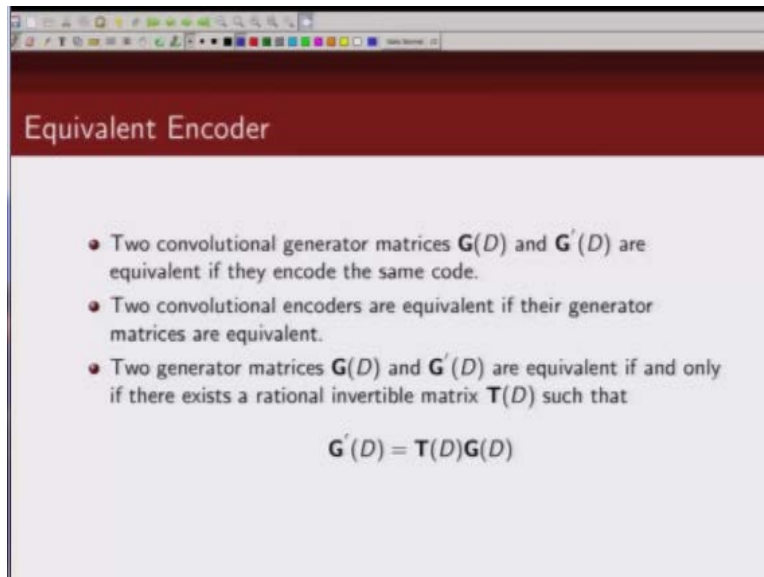
(Refer Slide Time: 15:02)



The image shows a screenshot of a presentation slide. The slide has a dark red header with the title "Equivalent Encoder" in white text. Below the header, the slide content is on a light gray background. It contains two bullet points, each starting with a red circular marker. The first bullet point states: "Two convolutional generator matrices  $\mathbf{G}(D)$  and  $\mathbf{G}'(D)$  are equivalent if they encode the same code." The second bullet point states: "Two convolutional encoders are equivalent if their generator matrices are equivalent." The slide is displayed within a window that has a standard operating system taskbar at the top.

Now we say do convolutional encoders are equivalent if their generator matrix are also equivalent in other words if they are a generator matrix encode the same code then we say two convolutional encoder are equivalent.

(Refer Slide Time: 15:24)



Equivalent Encoder

- Two convolutional generator matrices  $\mathbf{G}(D)$  and  $\mathbf{G}'(D)$  are equivalent if they encode the same code.
- Two convolutional encoders are equivalent if their generator matrices are equivalent.
- Two generator matrices  $\mathbf{G}(D)$  and  $\mathbf{G}'(D)$  are equivalent if and only if there exists a rational invertible matrix  $\mathbf{T}(D)$  such that

$$\mathbf{G}'(D) = \mathbf{T}(D)\mathbf{G}(D)$$

So if  $\mathbf{G}(D)$  and  $\mathbf{G}'(D)$  are equivalent this condition.

(Refer Slide Time: 15:33)

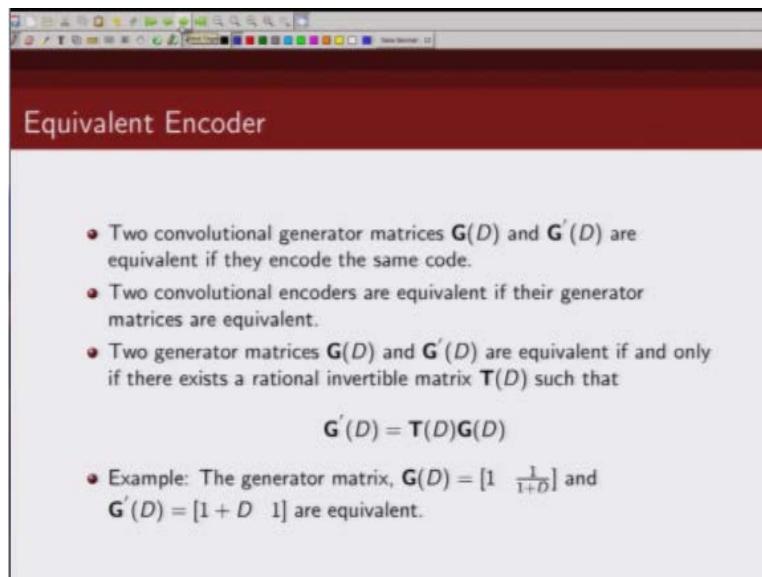
**Equivalent Encoder**

- Two convolutional generator matrices  $\mathbf{G}(D)$  and  $\mathbf{G}'(D)$  are equivalent if they encode the same code.
- Two convolutional encoders are equivalent if their generator matrices are equivalent.
- Two generator matrices  $\mathbf{G}(D)$  and  $\mathbf{G}'(D)$  are equivalent if and only if there exists a rational invertible matrix  $\mathbf{T}(D)$  such that

$$\underline{\mathbf{G}'(D) = \mathbf{T}(D)\mathbf{G}(D)} \quad \begin{aligned} v(D) &= u(D)\mathbf{G}'(D) \\ &= u(D)\mathbf{T}(D)\mathbf{G}(D) \\ &= u'(D)\mathbf{G}(D) \end{aligned}$$

Should hold so two generator matrix are equivalent if and only if there exists a rational invertible matrix  $\mathbf{T}(D)$  such that we can obtain  $\mathbf{G}'(D)$  by  $\mathbf{T}(D)$  multiplied by  $\mathbf{G}(D)$  okay and we can see basically so let us say set of codes generated by  $\mathbf{G}'(D)$  so that would be  $\mathbf{V}(D)$  it will be  $\mathbf{U}(D)$  times  $\mathbf{G}'(D)$  now this we can write as  $\mathbf{U}(D)\mathbf{T}(D)$  times  $\mathbf{G}(D)$  let us call  $\mathbf{U}(D)\mathbf{T}(D)$  is  $\mathbf{U}'(D)$   $\mathbf{G}(D)$  okay

(Refer Slide Time: 16:36)



**Equivalent Encoder**

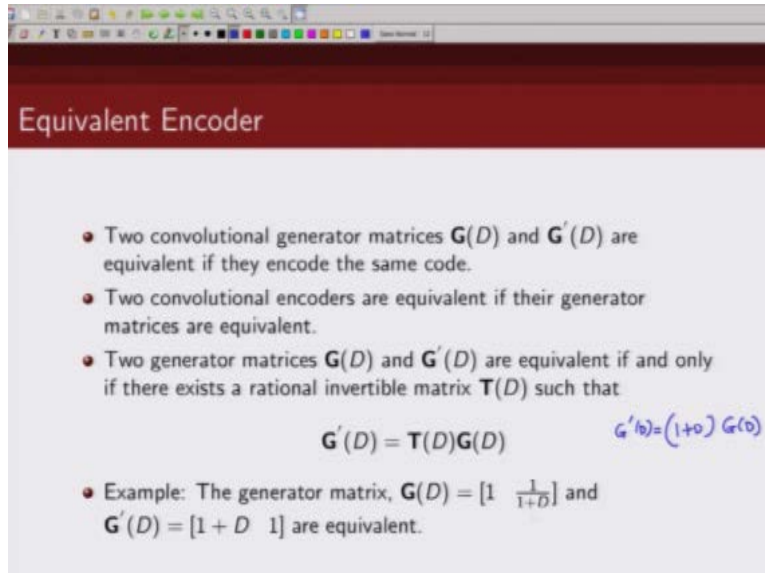
- Two convolutional generator matrices  $\mathbf{G}(D)$  and  $\mathbf{G}'(D)$  are equivalent if they encode the same code.
- Two convolutional encoders are equivalent if their generator matrices are equivalent.
- Two generator matrices  $\mathbf{G}(D)$  and  $\mathbf{G}'(D)$  are equivalent if and only if there exists a rational invertible matrix  $\mathbf{T}(D)$  such that

$$\mathbf{G}'(D) = \mathbf{T}(D)\mathbf{G}(D)$$

- Example: The generator matrix,  $\mathbf{G}(D) = \begin{bmatrix} 1 & \frac{1}{1+D} \end{bmatrix}$  and  $\mathbf{G}'(D) = \begin{bmatrix} 1+D & 1 \end{bmatrix}$  are equivalent.

So let us take an example this  $\mathbf{G}(D) = \begin{bmatrix} 1 & \frac{1}{1+D} \end{bmatrix}$  AND  $\mathbf{G}'(D) = \begin{bmatrix} 1+D & 1 \end{bmatrix}$  these are equivalent encoders because we can write

(Refer Slide Time: 16:55)



The image shows a presentation slide with a dark red header containing the title "Equivalent Encoder". Below the header, there are three bullet points. The third bullet point is followed by a mathematical equation and a handwritten note. The equation is  $G'(D) = T(D)G(D)$ . The handwritten note is  $G'(D) = (1+D)G(D)$ . Below the equation, there is another bullet point providing an example of equivalent generator matrices.

Equivalent Encoder

- Two convolutional generator matrices  $\mathbf{G}(D)$  and  $\mathbf{G}'(D)$  are equivalent if they encode the same code.
- Two convolutional encoders are equivalent if their generator matrices are equivalent.
- Two generator matrices  $\mathbf{G}(D)$  and  $\mathbf{G}'(D)$  are equivalent if and only if there exists a rational invertible matrix  $\mathbf{T}(D)$  such that

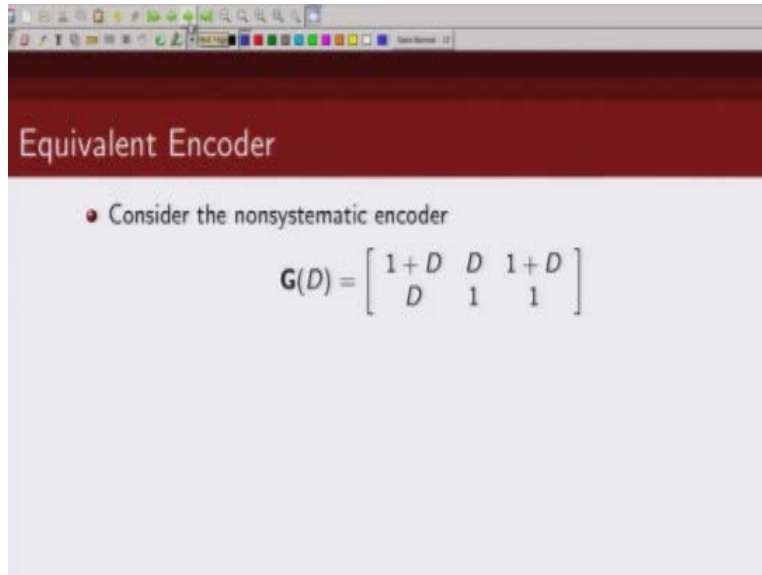
$$\mathbf{G}'(D) = \mathbf{T}(D)\mathbf{G}(D)$$

$G'(D) = (1+D)G(D)$

- Example: The generator matrix,  $\mathbf{G}(D) = [1 \quad \frac{1}{1+D}]$  and  $\mathbf{G}'(D) = [1+D \quad 1]$  are equivalent.

$G'(D)$  as  $1+D$  times  $G(D)$  okay so far and we can see this is a systematic encoder generator matrix for a systematic encoder okay now for.

(Refer Slide Time: 17:21)



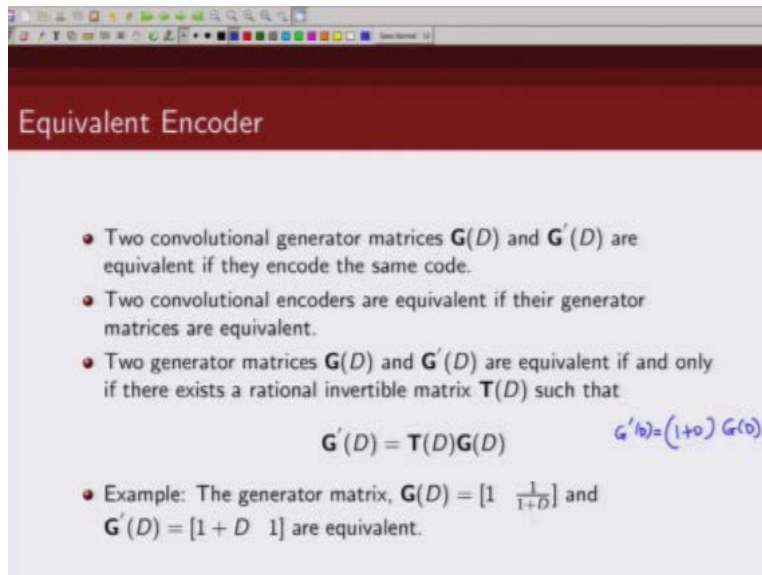
Equivalent Encoder

- Consider the nonsystematic encoder

$$\mathbf{G}(D) = \begin{bmatrix} 1+D & D & 1+D \\ D & 1 & 1 \end{bmatrix}$$

A and this is

(Refer Slide Time: 17:21)



Equivalent Encoder

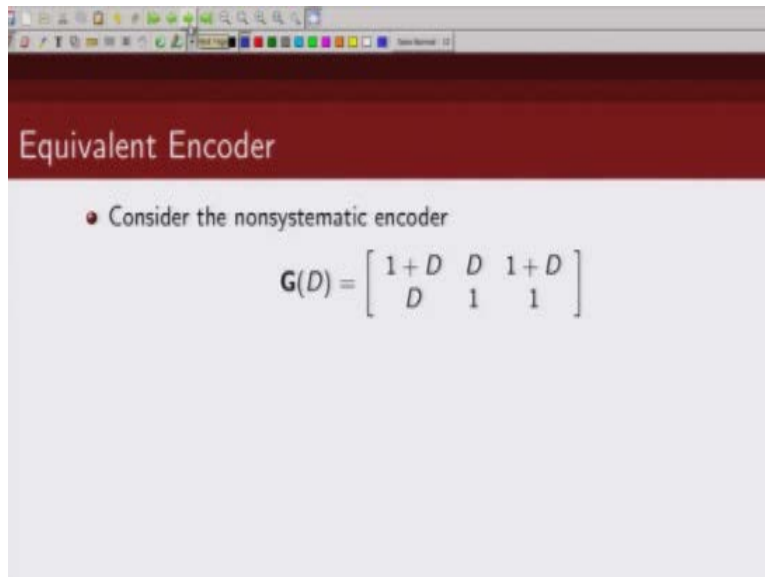
- Two convolutional generator matrices  $\mathbf{G}(D)$  and  $\mathbf{G}'(D)$  are equivalent if they encode the same code.
- Two convolutional encoders are equivalent if their generator matrices are equivalent.
- Two generator matrices  $\mathbf{G}(D)$  and  $\mathbf{G}'(D)$  are equivalent if and only if there exists a rational invertible matrix  $\mathbf{T}(D)$  such that

$$\mathbf{G}'(D) = \mathbf{T}(D)\mathbf{G}(D) \quad \mathbf{G}'(D) = (1+D) \mathbf{G}(D)$$

- Example: The generator matrix,  $\mathbf{G}(D) = [1 \quad \frac{1}{1+D}]$  and  $\mathbf{G}'(D) = [1+D \quad 1]$  are equivalent.

A feedback encoder this is a feed forward encoder.

(Refer Slide Time: 17:21)



Equivalent Encoder

- Consider the nonsystematic encoder

$$\mathbf{G}(D) = \begin{bmatrix} 1+D & D & 1+D \\ D & 1 & 1 \end{bmatrix}$$

So let us take an example of nonsystematic feed forward encoder and let us try to find it is equivalent systematic encoder. What would be the equivalent systematic encoder corresponding to this nonsystematic encoder the generator matrix



(Refer Slide Time: 17:56)

Equivalent Encoder

- Consider the nonsystematic encoder

$$\mathbf{G}(D) = \begin{bmatrix} 1+D & D & 1+D \\ D & 1 & 1 \end{bmatrix}$$
$$\mathbf{G}'(D) = \begin{bmatrix} \mathbf{I} & \mathbf{P} \\ \mathbf{0} & \mathbf{-} \end{bmatrix}$$

$\mathbf{G}'(D)$  should be of the form identity and some matrix  $\mathbf{P}$  so what we want is basically we want this to be of the form  $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$  something here so we want to convert this into a form of this type okay so we will do elementary operation to bring this generator matrix into a generator matrix of this form.

(Refer Slide Time: 18:35)

Equivalent Encoder

- Consider the nonsystematic encoder

$$\mathbf{G}(D) = \begin{bmatrix} 1+D & D & 1+D \\ D & 1 & 1 \end{bmatrix}$$

- Step 1: Row 1  $\Rightarrow [1/(1+D)][\text{Row 1}]$ .

$$\mathbf{G}_1(D) = \begin{bmatrix} 1 & D/(1+D) & 1 \\ D & 1 & 1 \end{bmatrix}$$

So first so we will do this transformation row 1 we will try to make this as 1 how can we make this as 1 if we multiply row 1 by  $1/(1+D)$  if we do that this term will become 1 this term will become  $D/(1+D)$  and this term will become 1 we leave this second row unchanged next

(Refer Slide Time: 19:06)

**Equivalent Encoder**

- Consider the nonsystematic encoder
 
$$\mathbf{G}(D) = \begin{bmatrix} 1+D & D & 1+D \\ D & 1 & 1 \end{bmatrix}$$
- Step 1: Row 1  $\Rightarrow [1/(1+D)][\text{Row 1}]$ .
 
$$\mathbf{G}_1(D) = \begin{bmatrix} 1 & D/(1+D) & 1 \\ D & 1 & 1 \end{bmatrix}$$
- Step 2: Row 2  $\Rightarrow \text{Row 2} + [D][\text{Row 1}]$ .
 
$$\mathbf{G}_2(D) = \begin{bmatrix} 1 & D/(1+D) & 1 \\ 0 & (1+D+D^2)/(1+D) & 1+D \end{bmatrix}$$

We want to get a 0 here right how do we get a 0 here we do this transformation row 2 will make it row 2 + D times row 1 so the first row is unchanged but second row we do this transformation it is row 2+ D times row 1 so row 2 here is D + D times row 1 which is another D so D + D is 0 similarly row 2 this  $1 + D^2 / 1+ D$  this is basically given by.

(Refer Slide Time: 19:51)

Equivalent Encoder

- Consider the nonsystematic encoder
$$\mathbf{G}(D) = \begin{bmatrix} 1+D & D & 1+D \\ D & 1 & 1 \end{bmatrix}$$
- Step 1: Row 1  $\Rightarrow$   $[1/(1+D)]$ [Row 1].
$$\mathbf{G}_1(D) = \begin{bmatrix} 1 & D/(1+D) & 1 \\ D & 1 & 1 \end{bmatrix}$$
- Step 2: Row 2  $\Rightarrow$  Row 2 +  $[D]$ [Row 1].
$$\mathbf{G}_2(D) = \begin{bmatrix} 1 & D/(1+D) & 1 \\ 0 & (1+D+D^2)/(1+D) & 1+D \end{bmatrix}$$

This and we have  $1+D$  which is this term so what we have done is we have converted this into form  $10$  next we want to get a  $1$  here right we want to get a  $1$  here so how can we get a  $1$  here we will.

(Refer Slide Time: 20:18)

Equivalent Encoder

- Consider the nonsystematic encoder
$$\mathbf{G}(D) = \begin{bmatrix} 1+D & D & 1+D \\ D & 1 & 1 \end{bmatrix}$$
- Step 1: Row 1  $\Rightarrow$   $[1/(1+D)]$ [Row 1].
$$\mathbf{G}_1(D) = \begin{bmatrix} 1 & D/(1+D) & 1 \\ D & 1 & 1 \end{bmatrix}$$
- Step 2: Row 2  $\Rightarrow$  Row 2 +  $[D]$ [Row 1].
$$\mathbf{G}_2(D) = \begin{bmatrix} 1 & D/(1+D) & 1 \\ 0 & (1+D+D^2)/(1+D) & 1+D \end{bmatrix}$$
- Step 3: Row 2  $\Rightarrow$   $[(1+D)/(1+D+D^2)]$ [Row 2].
$$\mathbf{G}_3(D) = \begin{bmatrix} 1 & D/(1+D) & 1 \\ 0 & 1 & (1+D^2)/(1+D+D^2) \end{bmatrix}$$

Do this following transformation so for row 2 we will multiply row 2 by  $1+D/1+D+D^2$  so if we do that then this will become 1 so we leave the first row uncharged here 0 if you multiple by this it does not change if you multiply this by this we get a one here and here we get this term so now what we have got so far is we got 1 here we got a 0 here we got a 1 here now what else is remaining we have to make this a.

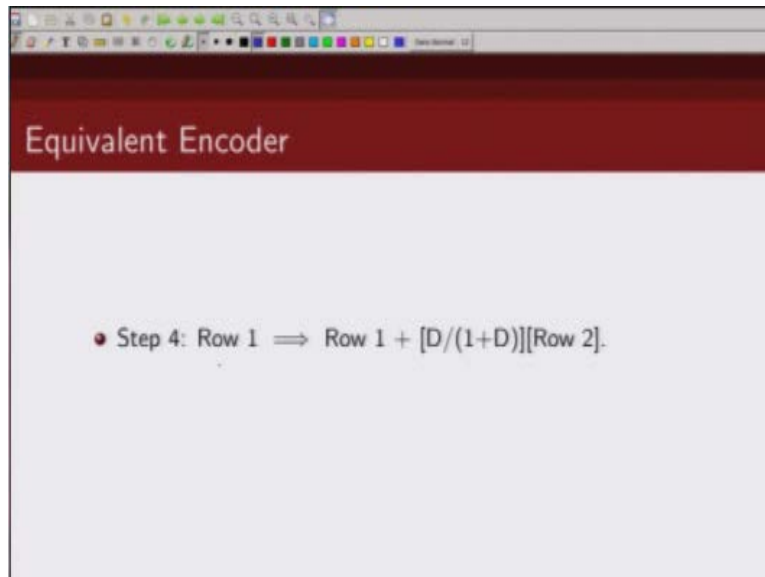
(Refer Slide Time: 20:58)

**Equivalent Encoder**

- Consider the nonsystematic encoder
 
$$\mathbf{G}(D) = \begin{bmatrix} 1+D & D & 1+D \\ D & 1 & 1 \end{bmatrix}$$
- Step 1: Row 1  $\Rightarrow [1/(1+D)][\text{Row 1}]$ .
 
$$\mathbf{G}_1(D) = \begin{bmatrix} 1 & D/(1+D) & 1 \\ D & 1 & 1 \end{bmatrix}$$
- Step 2: Row 2  $\Rightarrow \text{Row 2} + [D][\text{Row 1}]$ .
 
$$\mathbf{G}_2(D) = \begin{bmatrix} 1 & D/(1+D) & 1 \\ 0 & (1+D+D^2)/(1+D) & 1+D \end{bmatrix}$$
- Step 3: Row 2  $\Rightarrow [(1+D)/(1+D+D^2)][\text{Row 2}]$ .
 
$$\mathbf{G}_3(D) = \begin{bmatrix} 1 & D/(1+D) & 1 \\ 0 & 1 & (1+D^2)/(1+D+D^2) \end{bmatrix}$$

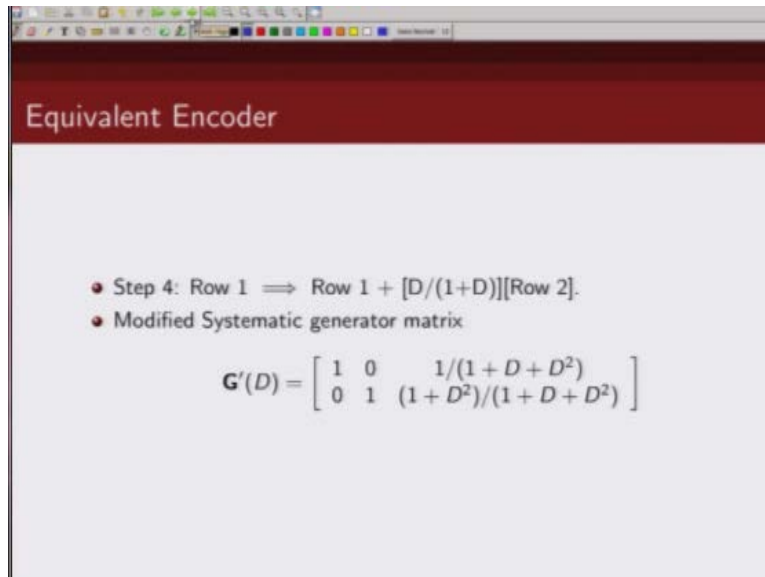
We make this a identity matrix so we have to make this as 0 how can we make this as 0 we multiply this by this and added up to the first row we can make it as 0 so next.

(Refer Slide Time: 21:16)



To row 1 we add  $D$  times  $\frac{1}{1+D}$  times row 2 if we do that

(Refer Slide Time: 21:25)



Equivalent Encoder

- Step 4: Row 1  $\implies$  Row 1 +  $[D/(1+D)]$ [Row 2].
- Modified Systematic generator matrix

$$\mathbf{G}'(D) = \begin{bmatrix} 1 & 0 & 1/(1+D+D^2) \\ 0 & 1 & (1+D^2)/(1+D+D^2) \end{bmatrix}$$

The modified generator matrix that we get is this note now this a generator matrix for a systematic encoder you have your identity matrix here

(Refer Slide Time: 21:37)

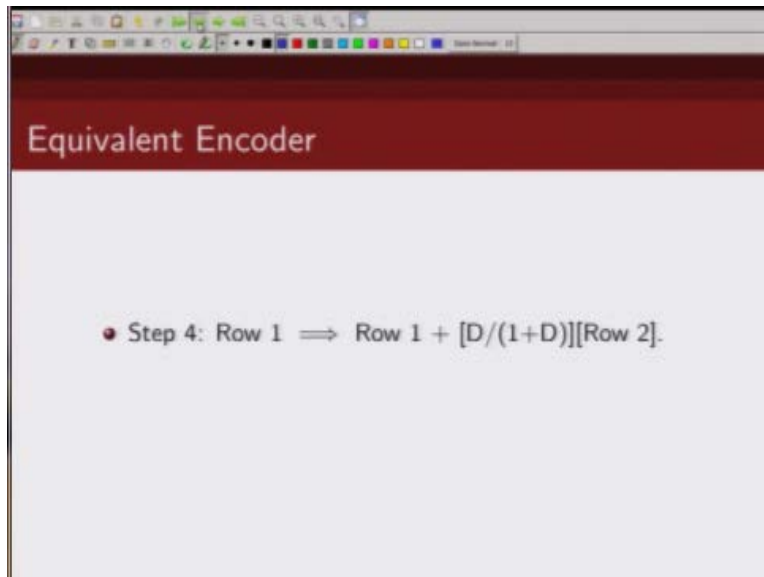
Equivalent Encoder

- Step 4: Row 1  $\implies$  Row 1 +  $[D/(1+D)]$ [Row 2].
- Modified Systematic generator matrix

$$G'(D) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1/(1+D+D^2) \\ (1+D^2)/(1+D+D^2) \end{bmatrix}$$

And you have some matrix here which is your P matrix so this is basically the generator matrix for a systematic encoder.

(Refer Slide Time: 21:53)



So note now.



(Refer Slide Time: 21:54)

**Equivalent Encoder**

- Consider the nonsystematic encoder
$$\mathbf{G}(D) = \begin{bmatrix} 1+D & D & 1+D \\ D & 1 & 1 \end{bmatrix}$$
- Step 1: Row 1  $\Rightarrow [1/(1+D)]$ [Row 1].
$$\mathbf{G}_1(D) = \begin{bmatrix} 1 & D/(1+D) & 1 \\ D & 1 & 1 \end{bmatrix}$$
- Step 2: Row 2  $\Rightarrow$  Row 2 + [D][Row 1].
$$\mathbf{G}_2(D) = \begin{bmatrix} 1 & D/(1+D) & 1 \\ 0 & (1+D+D^2)/(1+D) & 1+D \end{bmatrix}$$
- Step 3: Row 2  $\Rightarrow [(1+D)/(1+D+D^2)]$ [Row 2].
$$\mathbf{G}_3(D) = \begin{bmatrix} 1 & D/(1+D) & 1 \\ 0 & 1 & (1+D^2)/(1+D+D^2) \end{bmatrix}$$

By simple row operations we were able to get an equivalent systematic generator matrix for a none systematic encoder

(Refer Slide Time: 21:54)

### Equivalent Encoder

- Consider the nonsystematic encoder

$$\mathbf{G}(D) = \begin{bmatrix} 1+D & D & 1+D \\ D & 1 & 1 \end{bmatrix}$$

- Step 1: Row 1  $\Rightarrow$   $[1/(1+D)]$ [Row 1].

$$\mathbf{G}_1(D) = \begin{bmatrix} 1 & D/(1+D) & 1 \\ D & 1 & 1 \end{bmatrix}$$

- Step 2: Row 2  $\Rightarrow$  Row 2 +  $[D]$ [Row 1].

$$\mathbf{G}_2(D) = \begin{bmatrix} 1 & D/(1+D) & 1 \\ 0 & (1+D+D^2)/(1+D) & 1+D \end{bmatrix}$$

- Step 3: Row 2  $\Rightarrow$   $[(1+D)/(1+D+D^2)]$ [Row 2].

$$\mathbf{G}_3(D) = \begin{bmatrix} 1 & D/(1+D) & 1 \\ 0 & 1 & (1+D^2)/(1+D+D^2) \end{bmatrix}$$

By simple row operations we were able to get an equivalent systematic generator matrix for a nonsystematic encoder whose generator matrix is given by this.

(Refer Slide Time: 22:08)

## Catastrophic Encoder

- A convolutional encoder is catastrophic if it encodes some information sequence with infinitely many non-zero symbols into a code sequence with finitely many non-zero symbols.

Next we will explain the concept of catastrophic encoder, so convolutional encoder is catastrophic if it encodes some information sequence which has large weight which has large number of ones into a code sequence with finite number of ones so if you have an informational sequence let us say  $u(D)$  which is  $1/1+D$ .

(Refer Slide Time: 22:36)

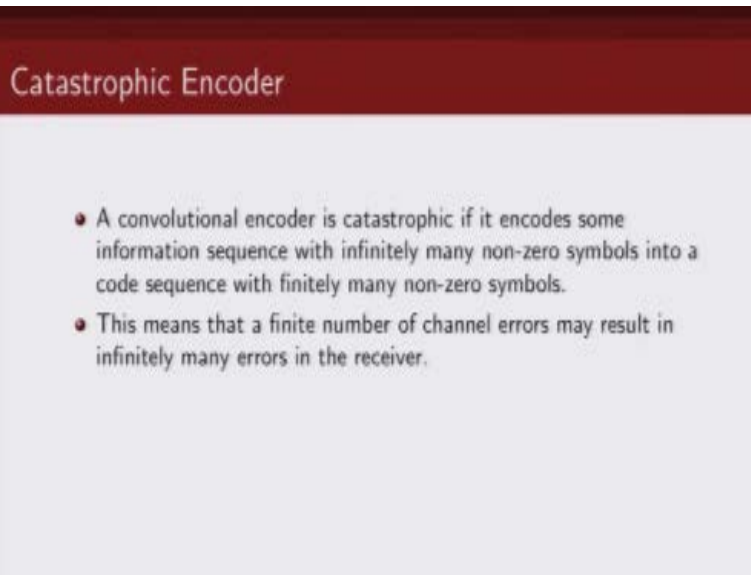
## Catastrophic Encoder

- A convolutional encoder is catastrophic if it encodes some information sequence with infinitely many non-zero symbols into a code sequence with finitely many non-zero symbols.

$$u(D) = \frac{1}{1+D}$$
$$= 1 + D + D^2 + \dots$$

Now this is a sequence of all ones, this is basically nothing but  $1 + D + D^2 + \dots$  so this is a sequence of one all ones. Now if you have an encoder which maps a sequence input sequence which has large number of ones into a sequence coded sequence with finite number of ones now that type of encoder is known as catastrophic, catastrophic encoder.

(Refer Slide Time: 23:11)



### Catastrophic Encoder

- A convolutional encoder is catastrophic if it encodes some information sequence with infinitely many non-zero symbols into a code sequence with finitely many non-zero symbols.
- This means that a finite number of channel errors may result in infinitely many errors in the receiver.

Now why is it catastrophic? So to illustrate it we will take an example it is a catastrophic encoder because a finite number of channel errors can result in infinite number of input errors because you had well inform just sequence which has large number of once possibly infinite number of once because that information sequences getting map to a coded sequence with finite number of once if error happens in those locations.

Where you have finite number of once then your output sequence will get transformed into an all zero sequence and your decoder will think that you have transmitted and all zero sequence, whereas actually you where transmitted a sequence of all once so finite number of channel errors in case of a catastrophic encoder can result in infinite number of input errors.

(Refer Slide Time: 24:11)

### Catastrophic Encoder

- A convolutional encoder is catastrophic if it encodes some information sequence with infinitely many non-zero symbols into a code sequence with finitely many non-zero symbols.
- This means that a finite number of channel errors may result in infinitely many errors in the receiver.
- Example:  
$$\mathbf{G}(D) = [1 + D \quad 1 + D^2]$$

Let us take an example of this encoder with generator matrix  $\mathbf{G}(D)$  which is given by  $1+D$  and  $1+D^2$  and let us feed input which is all sequence of all ones which I can write as  $1/1+D$ .

(Refer Slide Time: 24:26)

### Catastrophic Encoder

- A convolutional encoder is catastrophic if it encodes some information sequence with infinitely many non-zero symbols into a code sequence with finitely many non-zero symbols.
- This means that a finite number of channel errors may result in infinitely many errors in the receiver.
- Example:  
$$\mathbf{G}(D) = [1 + D \quad 1 + D^2] \quad u(D) = \frac{1}{1+D}$$

Now if this information sequence passes through this encoder what would be your output sequence it output sequence would be 1 and this will be 1+D.

(Refer Slide Time: 24:40)

### Catastrophic Encoder

- A convolutional encoder is catastrophic if it encodes some information sequence with infinitely many non-zero symbols into a code sequence with finitely many non-zero symbols.
- This means that a finite number of channel errors may result in infinitely many errors in the receiver.
- Example:

$$G(D) = [1 + D \quad 1 + D^2]$$
$$v(D) = [1 \quad 1 + D]$$
$$u(D) = \frac{1}{1 + D}$$

So what you will get is, you will get an output sequence which has weight only three, whereas your information sequence has infinite number of ones so here is an example where an input sequence of very large number of ones getting mapped to an output sequence of only weight three, what if error happens in these three locations where you had ones? Then your output sequence that the decoder will receive will be all zero sequence and the receiver will think that you transmitted all zero sequence.

(Refer Slide Time: 25:34)

## Catastrophic Encoder

- A convolutional encoder is catastrophic if it encodes some information sequence with infinitely many non-zero symbols into a code sequence with finitely many non-zero symbols.
- This means that a finite number of channel errors may result in infinitely many errors in the receiver.
- Example:

$$\mathbf{G}(D) = \begin{bmatrix} 1 + D & 1 + D^2 \end{bmatrix} \quad v(D) = \frac{1}{1+D}$$
$$v(D) = \begin{bmatrix} 1 & 1+D \end{bmatrix}$$

Whereas the input is all one sequence, so you can see in case of a catastrophic encoder a finite number of error in this example only three errors can result in infinite number of errors input errors, okay.



(Refer Slide Time: 25:50)

## Catastrophic Encoder

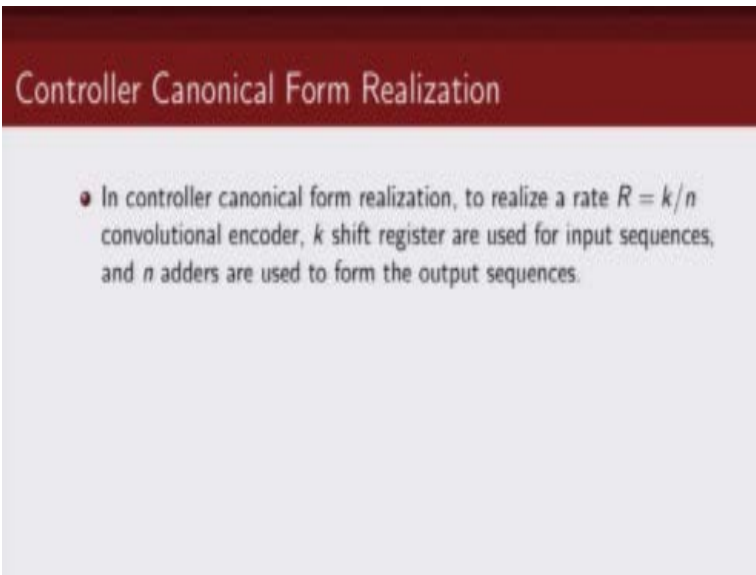
- A convolutional encoder is catastrophic if it encodes some information sequence with infinitely many non-zero symbols into a code sequence with finitely many non-zero symbols.
- This means that a finite number of channel errors may result in infinitely many errors in the receiver.
- Example:

$$\mathbf{G}(D) = [1 + D \quad 1 + D^2]$$

- If the input sequence  $\mathbf{u}(D) = \left[ \frac{1}{1+D} \right] = 1 + D + D^2 + \dots$ , then the output sequence,  $[1 \quad 1 + D]$  has only weight 3, even though the information sequence has infinite weight.

This I have explained.

(Refer Slide Time: 25:54)

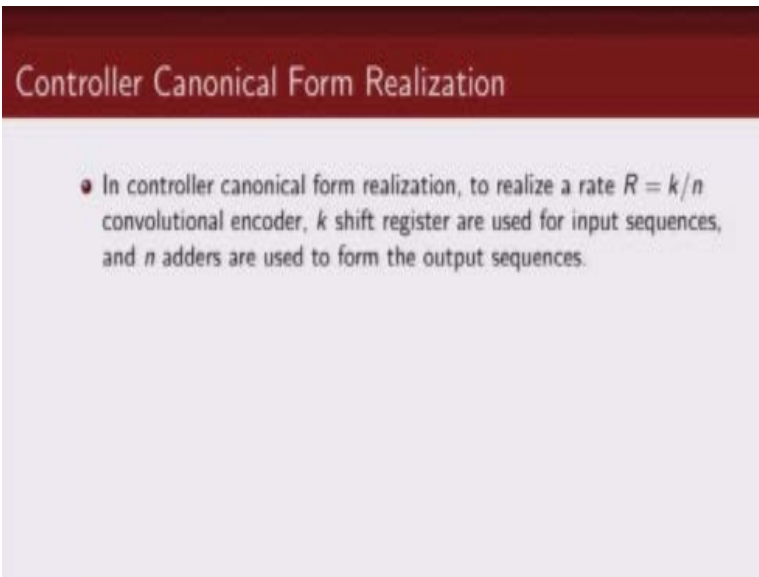


Controller Canonical Form Realization

- In controller canonical form realization, to realize a rate  $R = k/n$  convolutional encoder,  $k$  shift register are used for input sequences, and  $n$  adders are used to form the output sequences.

Next I am going to come to the topic of realization of a convolutional encoder how can we represent a convolutional encoder using shift register so given a generator matrix how can you implement a convolutional encoder, so in this we are going to talk about two such type of realization, the first one that we are going to discuss now is known as controller canonical form realization.

(Refer Slide Time: 26:27)

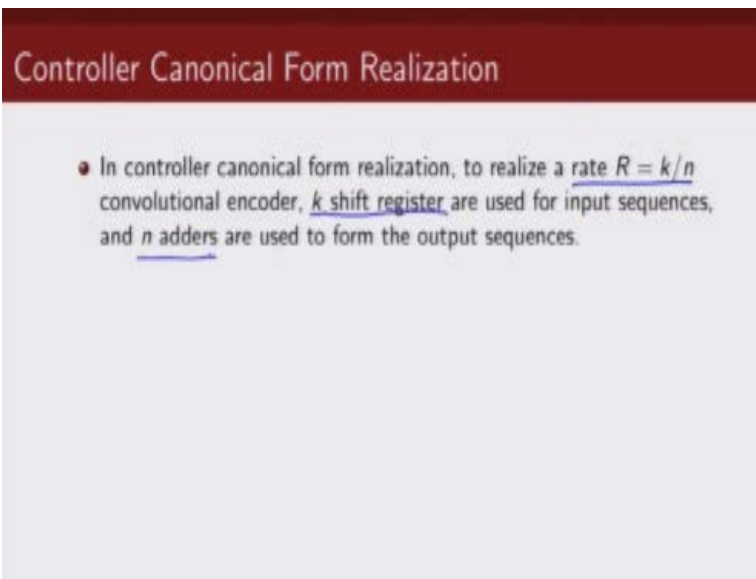


### Controller Canonical Form Realization

- In controller canonical form realization, to realize a rate  $R = k/n$  convolutional encoder,  $k$  shift registers are used for input sequences, and  $n$  adders are used to form the output sequences.

So in a controller canonical form realization if you have a rate  $R=k/n$  convolutional encoder we use  $k$  shift registers.

(Refer Slide Time: 26:30)



Controller Canonical Form Realization

- In controller canonical form realization, to realize a rate  $R = k/n$  convolutional encoder,  $k$  shift register are used for input sequences, and  $n$  adders are used to form the output sequences.

So the number of shift registers used is equal to number of information sequence that you have and the output is obtained by using  $n$  set of adders one for each output sequence.

(Refer Slide Time: 27:02)

## Controller Canonical Form Realization

- In controller canonical form realization, to realize a rate  $R = k/n$  convolutional encoder,  $k$  shift registers are used for input sequences, and  $n$  adders are used to form the output sequences.
- the  $k$  input sequences enter the shift registers at the left end of each shift register.

And In this case the  $k$  input sequences enter the shift register from the left and side and we take the output from the right and side.

(Refer Slide Time: 27:12)

## Controller Canonical Form Realization

- In controller canonical form realization, to realize a rate  $R = k/n$  convolutional encoder,  $k$  shift registers are used for input sequences, and  $n$  adders are used to form the output sequences.
- the  $k$  input sequences enter the shift registers at the left end of each shift register.
- The  $n$  adders used to obtain output sequences are external to the shift register.

The next point to remember here is in case of a controller canonical form realization these  $n$  adders that are used to obtain the output sequence the coded sequence these adders are external to the shift registers.

(Refer Slide Time: 27:29)

### Controller Canonical Form Realization

- In controller canonical form realization, to realize a rate  $R = k/n$  convolutional encoder,  $k$  shift registers are used for input sequences, and  $n$  adders are used to form the output sequences.
- the  $k$  input sequences enter the shift registers at the left end of each shift register.
- The  $n$  adders used to obtain output sequences are external to the shift register.

So they are not inside the shift registers.

(Refer Slide Time: 27:33)

### Controller Canonical Form Realization

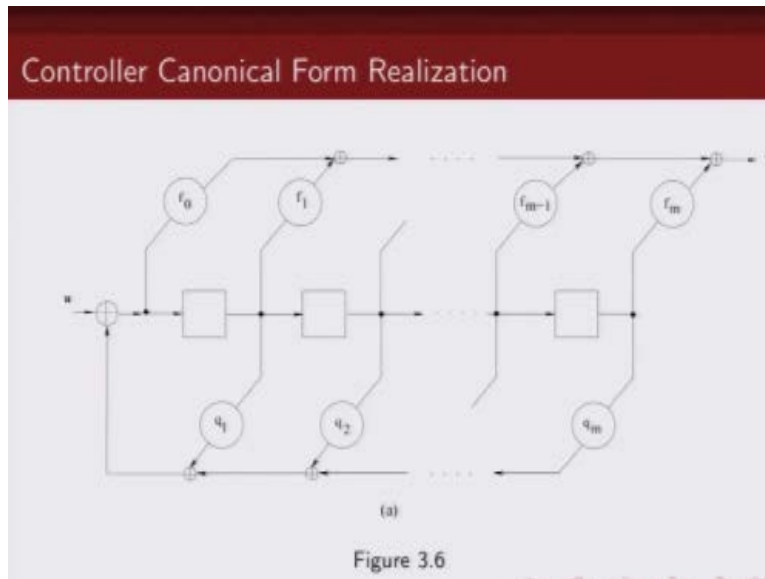
- In controller canonical form realization, to realize a rate  $R = k/n$  convolutional encoder,  $k$  shift registers are used for input sequences, and  $n$  adders are used to form the output sequences.
- the  $k$  input sequences enter the shift registers at the left end of each shift register.
- The  $n$  adders used to obtain output sequences are external to the shift register.
- In Figure 3.6(a) (next page), a rate  $R = 1$ , nonsystematic convolutional encoder with following generator function  $\mathbf{G}(D)$  is implemented in controller canonical form realization.

$$\mathbf{G}(D) = \left[ \frac{f_0 + f_1 D + \dots + f_{m-1} D^{m-1} + f_m D^m}{1 + q_1 D + q_2 D^2 + \dots + q_m D^m} \right]$$

So let us take an example for rate one in all systematic convolutional encoder whose generator matrix is given by this, so in the numerator you have  $f_0 + f_1 D + f_2 D^2$  like that similarly denominator you have  $1 + q_1 D + q_2 D^2$  like that so how can we implement this using controller canonical form realization so let us go back so we are going to use  $k$  shift registers so this is a rate one  $1/1$  so there will be only one shift register.



(Refer Slide Time: 28:14)



So we use one set of shift register corresponding to one input sequence, next.

(Refer Slide Time: 28:23)

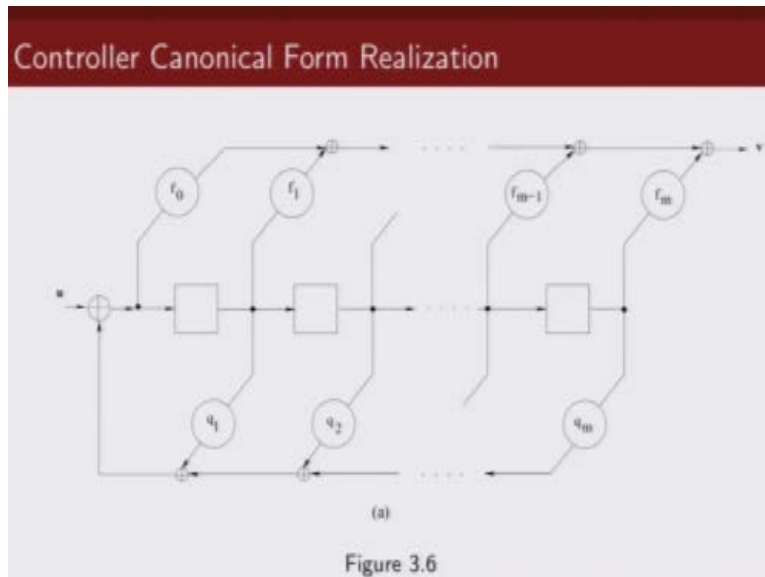
### Controller Canonical Form Realization

- In controller canonical form realization, to realize a rate  $R = k/n$  convolutional encoder,  $k$  shift registers are used for input sequences, and  $n$  adders are used to form the output sequences.
- the  $k$  input sequences enter the shift registers at the left end of each shift register.
- The  $n$  adders used to obtain output sequences are external to the shift register.
- In Figure 3.6(a) (next page), a rate  $R = 1$ , nonsystematic convolutional encoder with following generator function  $\mathbf{G}(D)$  is implemented in controller canonical form realization.

$$\mathbf{G}(D) = \left[ \begin{array}{c} f_0 + f_1 D + \dots + f_{m-1} D^{m-1} + f_m D^m \\ 1 + q_1 D + q_2 D^2 + \dots + q_m D^m \end{array} \right]$$

We use  $n$  set of adders now what is  $n$  here? Because is a rate one so  $n$  is also one so we will use one set of adders.

(Refer Slide Time: 28:31)



And these set of adders basically this output that we are seeing we have this  $n$  set of adders that we are using to obtain this coded sequence  $v$ .

(Refer Slide Time: 28:45)

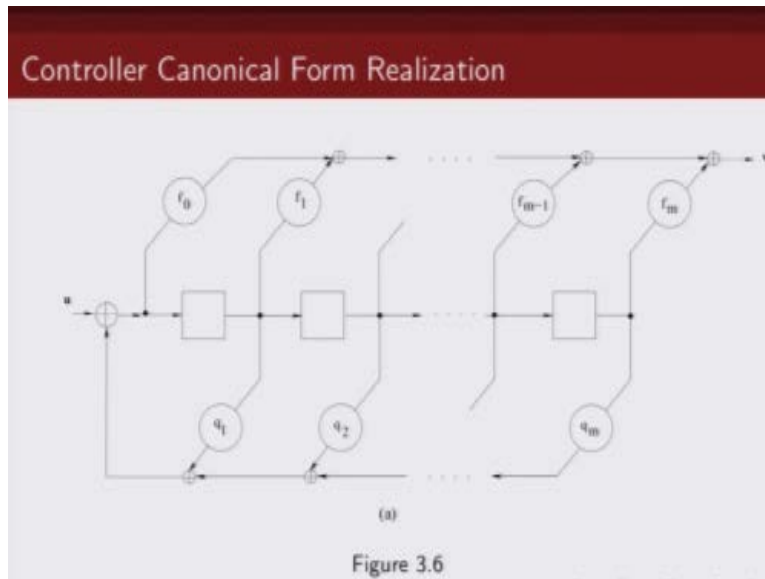
## Controller Canonical Form Realization

- In controller canonical form realization, to realize a rate  $R = k/n$  convolutional encoder,  $k$  shift registers are used for input sequences, and  $n$  adders are used to form the output sequences.
- the  $k$  input sequences enter the shift registers at the left end of each shift register.
- The  $n$  adders used to obtain output sequences are external to the shift register.
- In Figure 3.6(a) (next page), a rate  $R = 1$ , nonsystematic convolutional encoder with following generator function  $\mathbf{G}(D)$  is implemented in controller canonical form realization.

$$\mathbf{G}(D) = \left[ \begin{array}{c} f_0 + f_1 D + \dots + f_{m-1} D^{m-1} + f_m D^m \\ 1 + q_1 D + q_2 D^2 + \dots + q_m D^m \end{array} \right]$$

Now the key input sequence enter the shift register from the left and side so we can see here.

(Refer Slide Time: 28:53)



The input is entering from distance  $i$ , so since this is feedback encoder so let us first look at the numerator term.

(Refer Slide Time: 28:59)

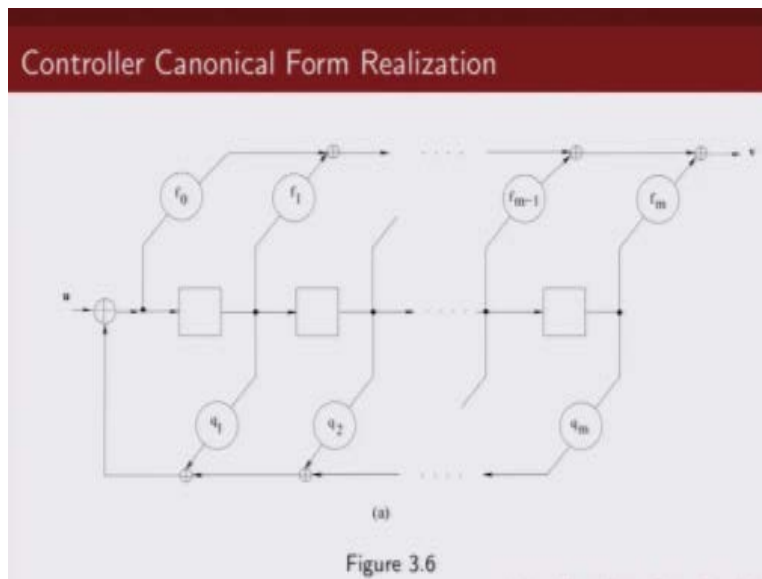
### Controller Canonical Form Realization

- In controller canonical form realization, to realize a rate  $R = k/n$  convolutional encoder,  $k$  shift registers are used for input sequences, and  $n$  adders are used to form the output sequences.
- the  $k$  input sequences enter the shift registers at the left end of each shift register.
- The  $n$  adders used to obtain output sequences are external to the shift register.
- In Figure 3.6(a) (next page), a rate  $R = 1$ , nonsystematic convolutional encoder with following generator function  $\mathbf{G}(D)$  is implemented in controller canonical form realization.

$$\mathbf{G}(D) = \left[ \frac{f_0 + f_1 D + \dots + f_{m-1} D^{m-1} + f_m D^m}{1 + q_1 D + q_2 D^2 + \dots + q_m D^m} \right]$$

What do we have here we have  $f_0 + f_1 D + f^2 D$  so this input is basically multiplied by  $f_0$ .

(Refer Slide Time: 22:15)



So the current input is getting multiplied by  $f_0$  then one delayed version of the input is getting multiplied by  $f_1$  two delayed version is multiplied by  $f_2$  so you can see this is an  $f_0$  this is  $f_1$  this is  $f_2$  and again whether there is a connection from this input to the output depending on that  $f_0$   $f_1$   $f_2$  will be either one or zero if there is a connection this will be one if there is no connection this will be zero. So you can see this, this is  $f_0$  this is  $f_1 D$ ,  $f_2 D^2$  like that basically if this is emit delay element this will be  $f_m D_m$ . Similarly you look go back and look at the denominator.

(Refer Slide Time: 30:06)

### Controller Canonical Form Realization

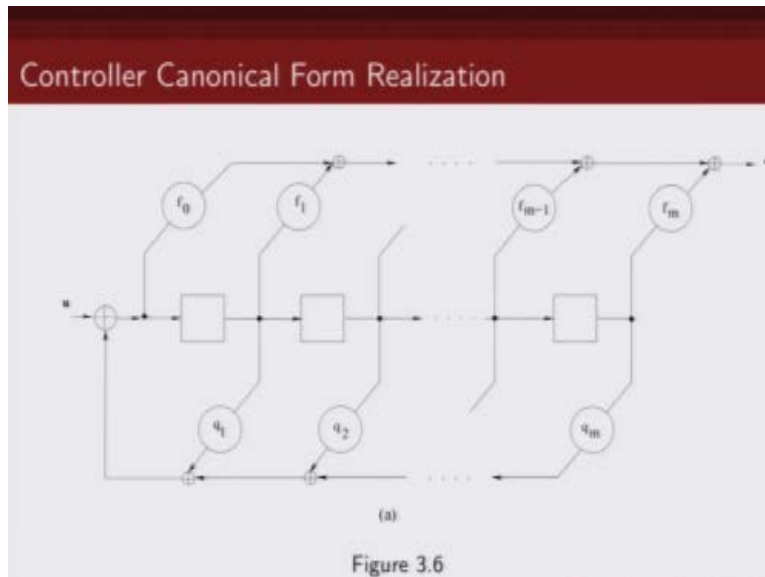
- In controller canonical form realization, to realize a rate  $R = k/n$  convolutional encoder,  $k$  shift registers are used for input sequences, and  $n$  adders are used to form the output sequences.
- the  $k$  input sequences enter the shift registers at the left end of each shift register.
- The  $n$  adders used to obtain output sequences are external to the shift register.
- In Figure 3.6(a) (next page), a rate  $R = 1$ , nonsystematic convolutional encoder with following generator function  $\mathbf{G}(D)$  is implemented in controller canonical form realization.

$$\mathbf{G}(D) = \left[ \frac{f_0 + f_1 D + \dots + f_{m-1} D^{m-1} + f_m D^m}{1 + q_1 D + q_2 D^2 + \dots + q_m D^m} \right]$$

We have  $1 + q_1 D + q_2 D^2$  like that.

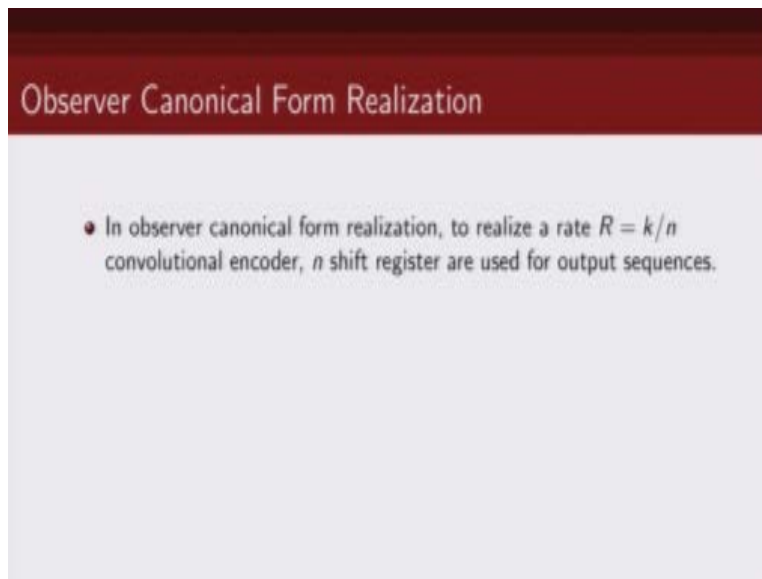


(Refer Slide Time: 30:14)



So this the input one this is the D term  $u_1$  D term so this is multiplied by  $q_1$  this is  $D^2$  term multiplied by  $q_2$  like that and then finally you have  $D_m$  term which is I am getting multiplied by  $q_m$ . So you can see this is how we can realize a convolutional code using controller canonical form realization please note these adders are external to the shift register so there is not adders here internal to shift registers. The inputs are entering on the left and side where the output is taken from right and side.

(Refer Slide Time: 31:01)

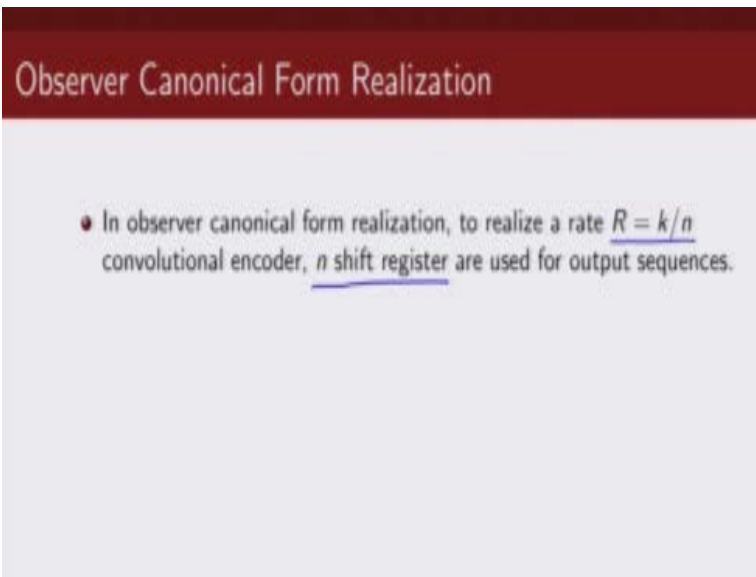


Observer Canonical Form Realization

- In observer canonical form realization, to realize a rate  $R = k/n$  convolutional encoder,  $n$  shift register are used for output sequences.

Now contrast is with observer canonical form realization so now observer canonical form realization we need to realize the rate  $k/n$  encoder we require  $n$  shift registers.

(Refer Slide Time: 31:11)

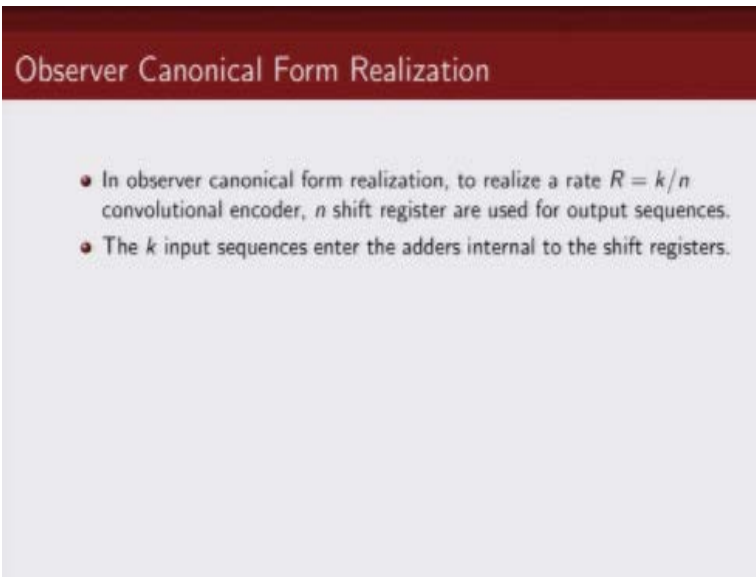


Observer Canonical Form Realization

- In observer canonical form realization, to realize a rate  $R = k/n$  convolutional encoder,  $n$  shift register are used for output sequences.

Now please note for the controller canonical form realization we required  $k$  set of shift registers whereas in this case we require  $n$  set of shift registers one for each of the coded bits.

(Refer Slide Time: 31:35)

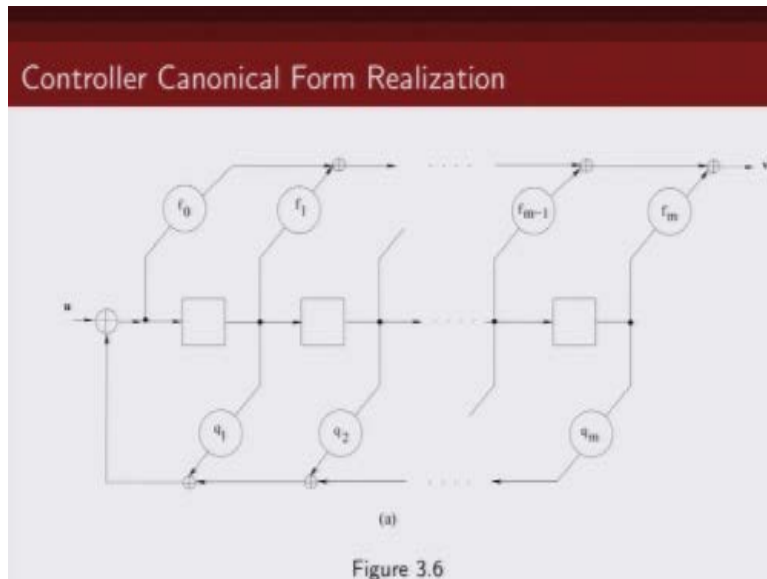


Observer Canonical Form Realization

- In observer canonical form realization, to realize a rate  $R = k/n$  convolutional encoder,  $n$  shift register are used for output sequences.
- The  $k$  input sequences enter the adders internal to the shift registers.

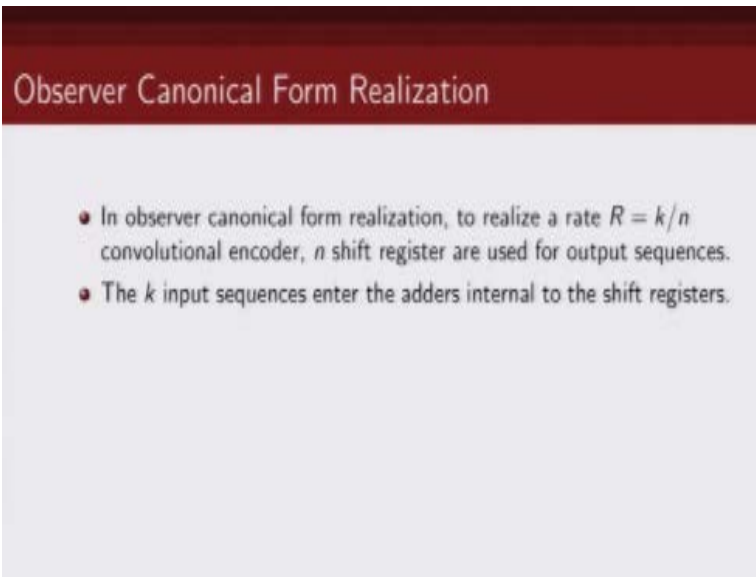
The second difference is the key input sequence is in the observer canonical form realization these key sequences enter into the shift register and these adders are internal to the shift register if you recall.

(Refer Slide Time: 31:56)



In case of a controller canonical form realization the input is entering here and a  $t$  time is still set when your clock comes they move they shift to one location to the right, this will move to here, this will move to here.

(Refer Slide Time: 32:11)



Observer Canonical Form Realization

- In observer canonical form realization, to realize a rate  $R = k/n$  convolutional encoder,  $n$  shift register are used for output sequences.
- The  $k$  input sequences enter the adders internal to the shift registers.

Whereas in the observer canonical form realization these inputs are directly entering into the shift register and these adders are internal to the shift register we will give an example to illustrate what we mean.

(Refer Slide Time: 32:25)

### Observer Canonical Form Realization

- In observer canonical form realization, to realize a rate  $R = k/n$  convolutional encoder,  $n$  shift register are used for output sequences.
- The  $k$  input sequences enter the adders internal to the shift registers.
- The lowest degree term in the generator polynomial represent the connections to the right hand side of the shift registers.

The lowest degree term generator matrix represent the connection to the right and side of the shift register, in case of controller canonical form realization the lowest degree term was on the left and side here the lowest degree term will be on the right and side.

(Refer Slide Time: 32:46)

## Observer Canonical Form Realization

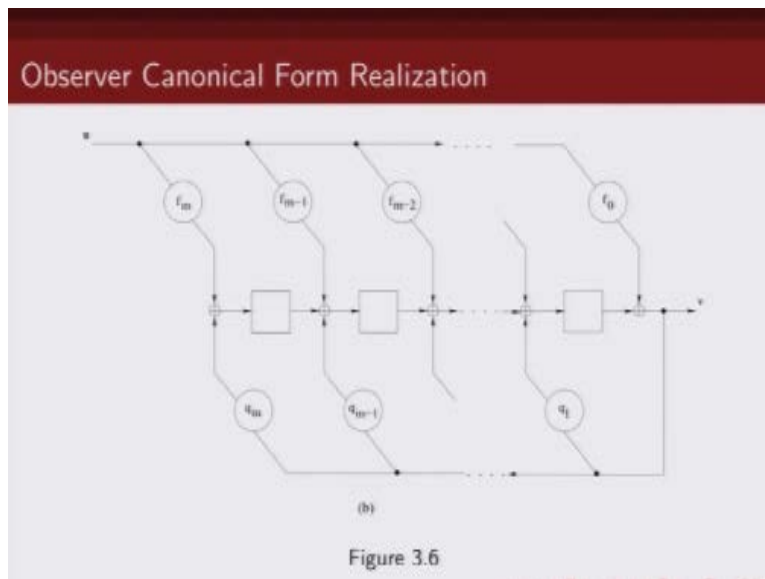
- In observer canonical form realization, to realize a rate  $R = k/n$  convolutional encoder,  $n$  shift registers are used for output sequences.
- The  $k$  input sequences enter the adders internal to the shift registers.
- The lowest degree term in the generator polynomial represent the connections to the right hand side of the shift registers.
- In Figure 3.6(b) (next page), a rate  $R = 1$ , nonsystematic convolutional encoder with following generator function  $\mathbf{G}(D)$  is implemented in observer canonical form realization.

$$\mathbf{G}(D) = \left[ \frac{f_0 + f_1 D + \dots + f_{m-1} D^{m-1} + f_m D^m}{1 + q_1 D + q_2 D^2 + \dots + q_m D^m} \right]$$

So let us take the same example that we considered earlier so we are considering the same generator matrix and we are going to realize this generator matrix now using observer canonical form realization. So again here  $k$  is one  $n$  is one so we have  $n$  is one so we have one set of shift register this is one set of shift registers.



(Refer Slide Time: 33:07)



Next what did we say?

(Refer Slide Time: 33:10)

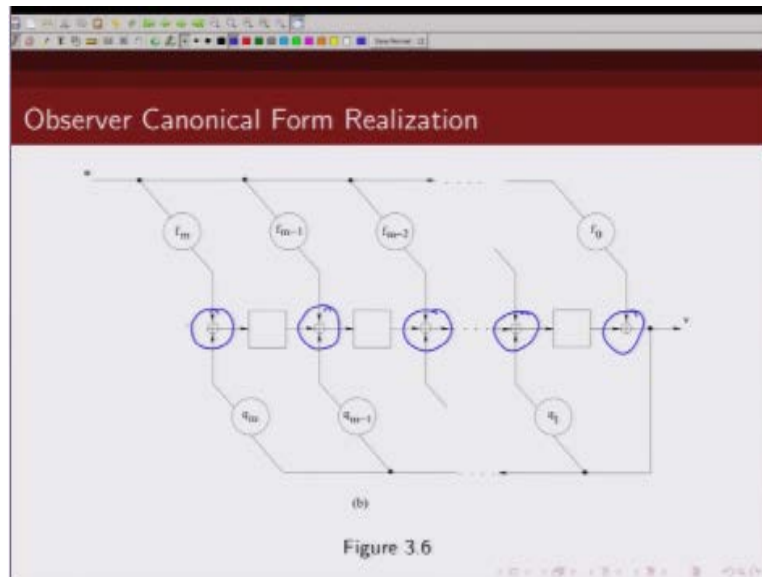
**Observer Canonical Form Realization**

- In observer canonical form realization, to realize a rate  $R = k/n$  convolutional encoder,  $n$  shift register are used for output sequences.
- The  $k$  input sequences enter the adders internal to the shift registers.
- The lowest degree term in the generator polynomial represent the connections to the right hand side of the shift registers.
- In Figure 3.6(b) (next page), a rate  $R = 1$ , nonsystematic convolutional encoder with following generator function  $\mathbf{G}(D)$  is implemented in observer canonical form realization.

$$\mathbf{G}(D) = \left[ \frac{f_0 + f_1 D + \dots + f_{m-1} D^{m-1} + f_m D^m}{1 + q_1 D + q_2 D^2 + \dots + q_m D^m} \right]$$

The key input sequence enter the adder internal to the shift register.

(Refer Slide Time: 33:20)



And what do we mean by internal. So these are the shift register elements delay elements, and no more these adders are in between the shift register these adders are internal to the shift registers okay.

(Refer Slide Time: 33:36)

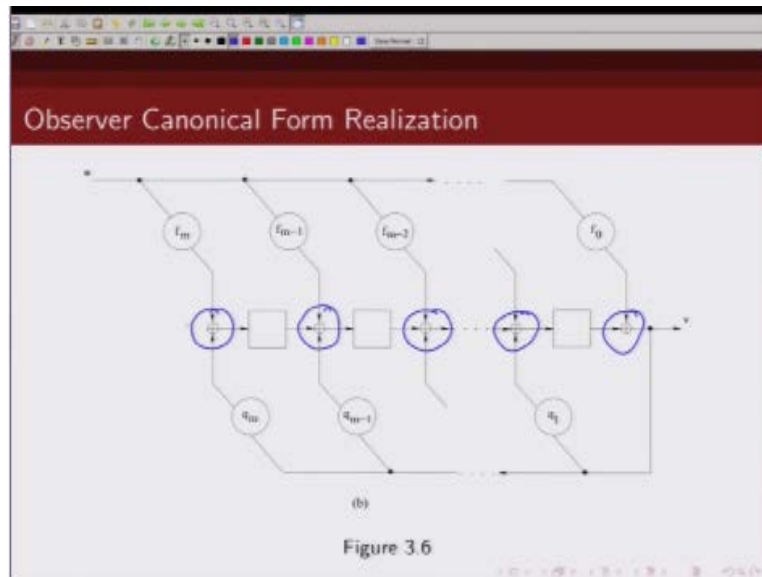
**Observer Canonical Form Realization**

- In observer canonical form realization, to realize a rate  $R = k/n$  convolutional encoder,  $n$  shift register are used for output sequences.
- The  $k$  input sequences enter the adders internal to the shift registers.
- The lowest degree term in the generator polynomial represent the connections to the right hand side of the shift registers.
- In Figure 3.6(b) (next page), a rate  $R = 1$ , nonsystematic convolutional encoder with following generator function  $\mathbf{G}(D)$  is implemented in observer canonical form realization.

$$\mathbf{G}(D) = \left[ \frac{f_0 + f_1 D + \dots + f_{m-1} D^{m-1} + f_m D^m}{1 + q_1 D + q_2 D^2 + \dots + q_m D^m} \right]$$

And next thing that we said was a lowest degree term in the generator matrix, represents connection to the right hand side.

(Refer Slide Time: 33:43)



You see here, the inputs are directly coming to the adder. So this term is corresponding to  $S^0$   $u(d)$  this term is corresponds to  $f_1 d$  of  $u(d)$ . Where as in the controller canonical form the left most term was  $f_0$  and the right most was  $f_m$ . Here, is just opposite, so you can see this is  $f_0$  term,  $f_1$  term,  $f_2$  term and similarly in the denominator you can see this is  $q_1$ , this is  $q_2$  like that this will be  $q_m$ . Same generator matrix can be realized using two different forms.

(Refer Slide Time: 34:32)

Realization of Convolutional encoder

Example 3.7:

- Let's consider a rate  $R = 2/3$  systematic feedforward encoder with generator matrix

$$\mathbf{G}(D) = \begin{bmatrix} 1 & 0 & 1+D+D^2 \\ 0 & 1 & 1+D \end{bmatrix}$$

- The parity check matrix can be written as

$$\mathbf{H}(D) = \begin{bmatrix} \mathbf{h}^{(0)}(D) & \mathbf{h}^{(1)}(D) & 1 \end{bmatrix} = \begin{bmatrix} 1+D+D^2 & 1+D & 1 \end{bmatrix}$$

- The controller canonical form realization results in  $(\overset{n}{3}, \overset{k}{2}, \overset{m}{3})$  encoder.

So let us take an example to illustrate this, so we are considering our rate 2/3 systematic feed forward encoder whose generator matrix is given by this. Now let us try to realize this generator matrix using controller canonical form realization and observer form realization. So the parity check matrix for this is given by this expression we will just show you that so in controller canonical form realization we have so there are two inputs here so we will have one set of shift register for each of the input.

So we will have one set of shift register for this and one set of shift register for this. And to realize this we need two memory elements because here the highest degree of D is 2. And to realize this we required one memory element.

So total we will require three memory elements so that is what I said for the controller canonical form realization for this rate two third this is my n, this is k, and this is the memory order. We basically require three memory elements to represent this convolutional encoder in the controller canonical form realization.

(Refer Slide Time: 36:14)

Realization of Convolutional encoder

Example 3.7:

- Let's consider a rate  $R = 2/3$  systematic feedforward encoder with generator matrix

$$\mathbf{G}(D) = \begin{bmatrix} 1 & 0 & 1 + D + D^2 \\ 0 & 1 & 1 + D \end{bmatrix}.$$

- The parity check matrix can be written as

$$\mathbf{H}(D) = [\mathbf{h}^{(0)}(D) \quad \mathbf{h}^{(1)}(D) \quad 1] = [1 + D + D^2 \quad 1 + D \quad 1].$$

- The controller canonical form realization results in (3, 2, 3) encoder.
- The observer canonical form realization results in (3, 2, 2) encoder.

Now what about observer canonical form realization in observer canonical form realization we use wise set of shift register for each of the end coded bits.

(Refer Slide Time: 36:28)

Realization of Convolutional encoder

Example 3.7:

- Let's consider a rate  $R = 2/3$  systematic feedforward encoder with generator matrix

$$\mathbf{G}(D) = \begin{bmatrix} 1 & 0 & 1 + D + D^2 \\ 0 & 1 & 1 + D \end{bmatrix}$$

- The parity check matrix can be written as

$$\mathbf{H}(D) = [\mathbf{h}^{(0)}(D) \quad \mathbf{h}^{(1)}(D) \quad 1] = [1 + D + D^2 \quad 1 + D \quad 1]$$

- The controller canonical form realization results in  $(3, 2, 3)$  encoder.
- The observer canonical form realization results in  $(3, 2, 2)$  encoder.

So how many coded bits we have, we have three, one is this, one is this and one is this. Now how many memory elements you required to represent this 0 is directly the input coming in here. Here, 0 because the direct input is coming here, and what about this its maximum degree is 2 so we will require 2. So over all for this generator matrix if we try to realize it using observer canonical form realization, we require only two memory elements. And in the next slide I am going to show you those two encoder realization.





(Refer Slide Time: 37:18)

Realization of Convolutional encoder

Example 3.7:

- Let's consider a rate  $R = 2/3$  systematic feedforward encoder with generator matrix

$$\mathbf{G}(D) = \begin{bmatrix} 1 & 0 & 1 + D + D^2 \\ 0 & 1 & 1 + D \end{bmatrix}$$

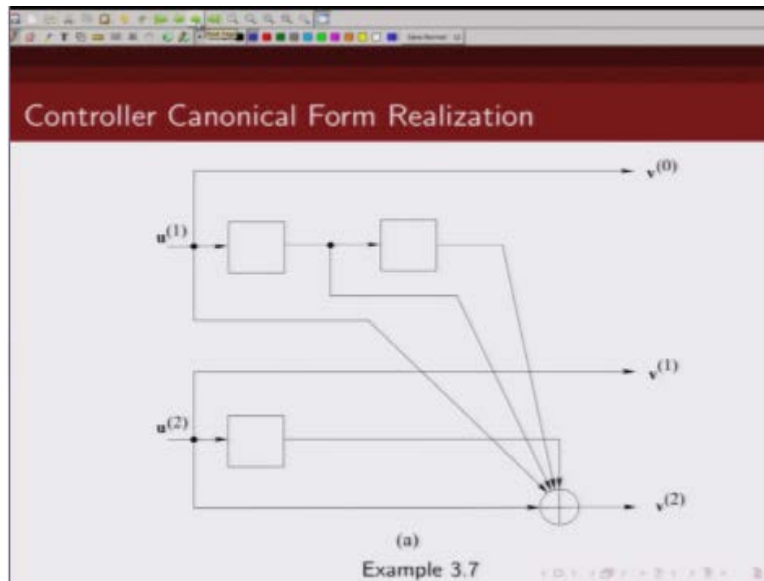
- The parity check matrix can be written as

$$\mathbf{H}(D) = [\mathbf{h}^{(0)}(D) \quad \mathbf{h}^{(1)}(D) \quad 1] = [1 + D + D^2 \quad 1 + D \quad 1]$$

- The controller canonical form realization results in  $(3, 2, 3)$  encoder.
- The observer canonical form realization results in  $(3, 2, 2)$  encoder.

So let me just write down the generator matrix.

(Refer Slide Time: 37:22)



So my generator matrix  $G(D)$  is 1001.

(Refer Slide Time: 37:28)

Realization of Convolutional encoder

Example 3.7:

- Let's consider a rate  $R = 2/3$  systematic feedforward encoder with generator matrix

$$\mathbf{G}(D) = \begin{bmatrix} 1 & 0 & 1 + D + D^2 \\ 0 & 1 & 1 + D \end{bmatrix}$$

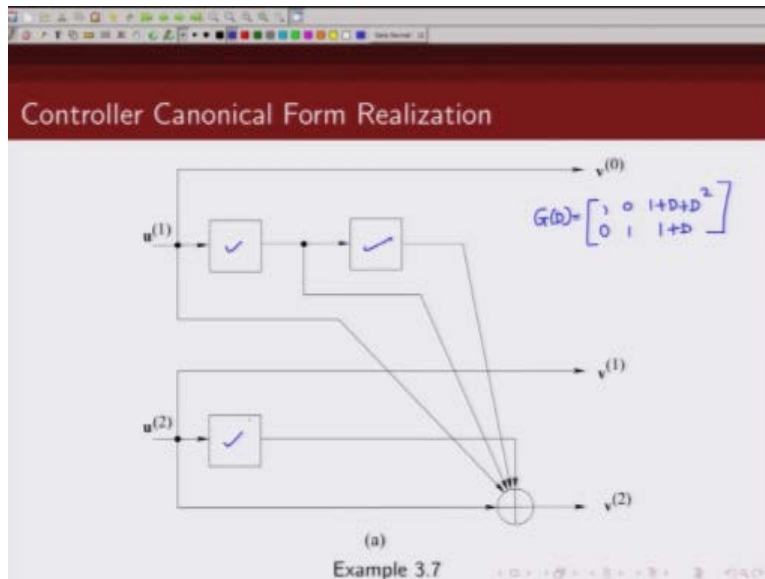
- The parity check matrix can be written as

$$\mathbf{H}(D) = \begin{bmatrix} \mathbf{h}^{(0)}(D) & \mathbf{h}^{(1)}(D) & 1 \end{bmatrix} = \begin{bmatrix} 1 + D + D^2 & 1 + D & 1 \end{bmatrix}$$

- The controller canonical form realization results in (3, 2, 3) encoder.
- The observer canonical form realization results in (3, 2, 2) encoder.

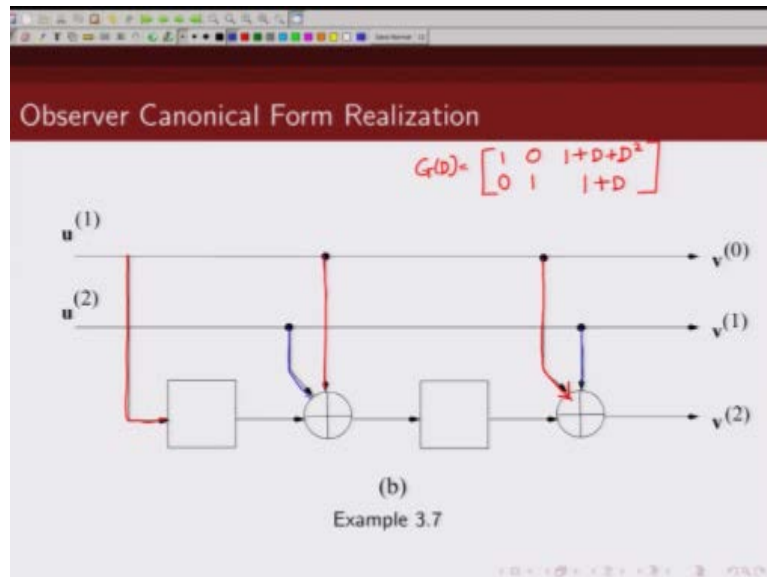
Then do we have  $1+D+D^2$ .

(Refer Slide Time: 37:32)



$1+D+D^2$ ,  $1+D$ . so one set of shift register for this, so this maximum degree is of this is 2 so we use two memory elements and for this maximum degree of  $D$  is 1 so we use one memory element okay. Now what is the first output first output is directly input  $u^{(1)}$  so this is my  $u^{(1)}$  second output is directly  $u^{(2)}$  this is basically this and the third output is  $1+D+D^2$  of  $u^{(1)}$  so this is  $u$ , this is the  $D$  of  $u$   $D$ ,  $u^{(1)}D$  and this is  $D^2$  of  $u^{(1)} D$  plus  $1+D$  times  $u$  to  $D$  so  $1+D$  meaning one term is  $u^{(2)}$  and second is delayed version of  $u^{(2)}$ . So this will be my third coded bit. So you can see to realize this generator matrix, we required total three memory elements one, two and three.

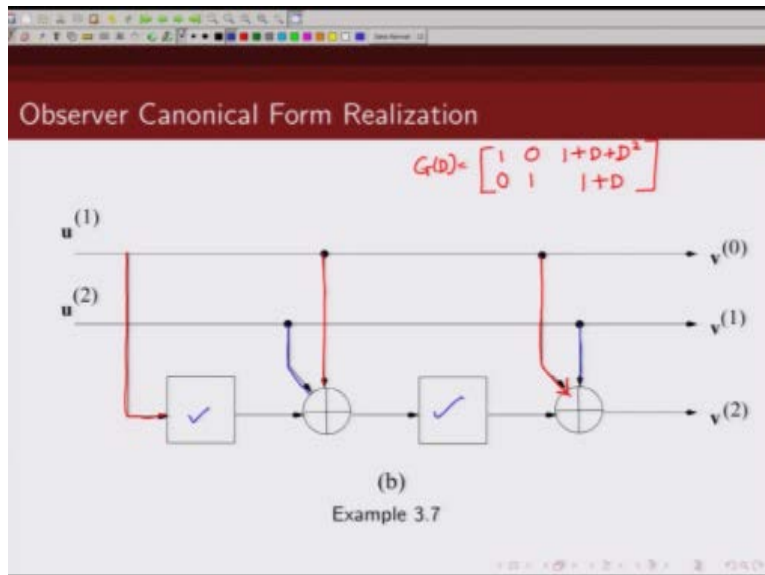
(Refer Slide Time: 39:03)



Now let us see for the observer canonical form realization, again let me write down my generator matrix this is  $1001, 1+D+D^2, 1+D$ . So we said one set of rows, one set of shift register for each of the coded bit for each column there is one set of shift register what some maximum delay element here 0 so you can see directly. What about this again there maximum degree of  $D$  is basically 0 so this no shift register. And here for the third line  $D$  is 2 so we took  $2D$ . So what is the final output then first one is first, first coded bit is just  $u^{(1)}$  of  $D$  that is what it is.

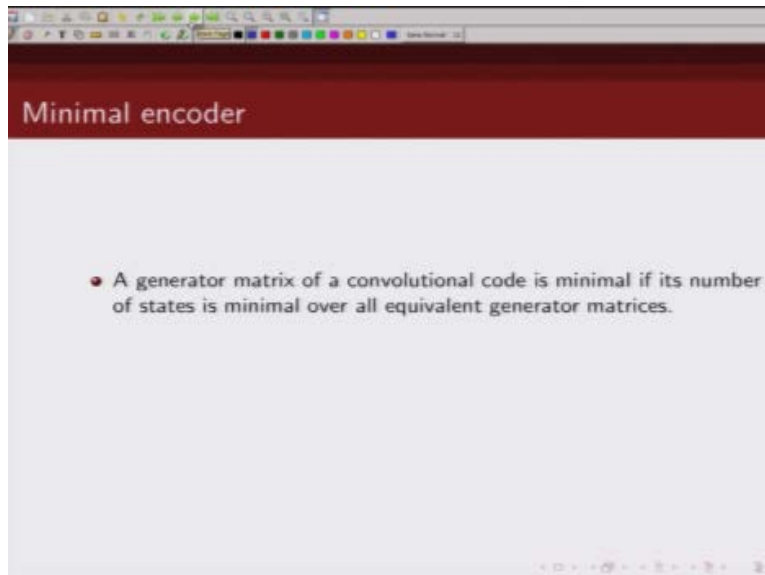
Second coded bit is  $u^{(2)}$  of  $D$  straight this and third coded bit is  $1+D+D^2$  of  $u^{(1)}$  of  $D$ . So what is  $u^{(1)}$  of  $D$ ,  $u^{(1)}$  of  $D$  is this one. What is  $D$  times  $u^{(1)}$  of  $D$  that is this term, and what is  $D^2$   $u^{(1)}$  of  $D$  that is this term. Find plus  $1+D$  times  $u^{(2)}$  of  $D$ . So then what we have is  $u^{(2)}$  of  $D$  is this and  $D$  times  $u^{(2)}$  of  $D$  is this. So this is our observer canonical form realization for this convolutional encoder with this generator matrix and note.

(Refer Slide Time: 41:07)



We only required 2, 1, 2 we required only two memory elements. So same encoder here require two memory elements for the controller canonical form realization we required three memory elements.

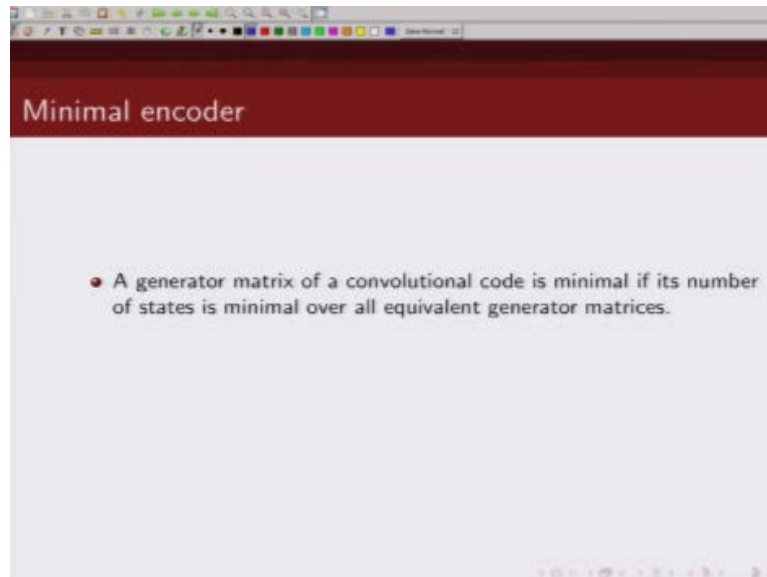
(Refer Slide Time: 41:24)



So that brings us to this notion of minimal encoder we saw the same encode convolutional encoder with same generator matrix can be realized using two different ways one that resulted in three memory elements other that resulted in two memory elements.

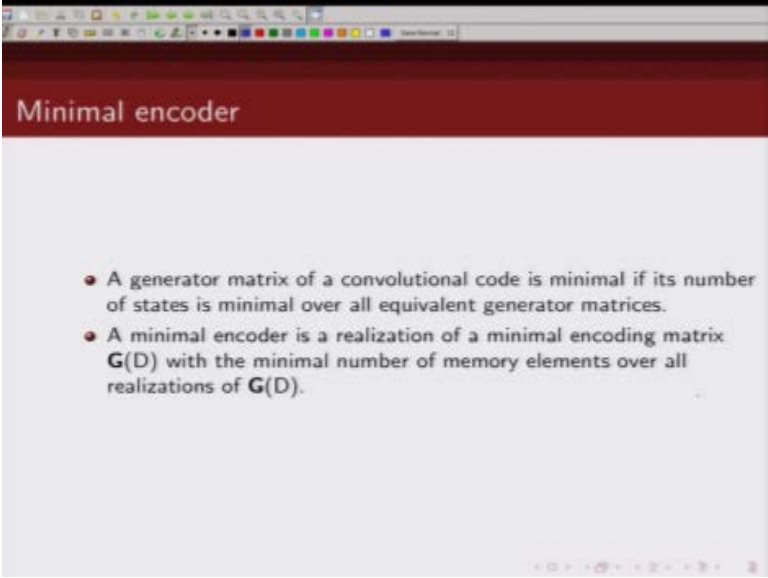


(Refer Slide Time: 41:43)



So we say a generator matrix is minimal if the, if its number of states is minimal over all possible equivalent generator matrix.

(Refer Slide Time: 41:54)

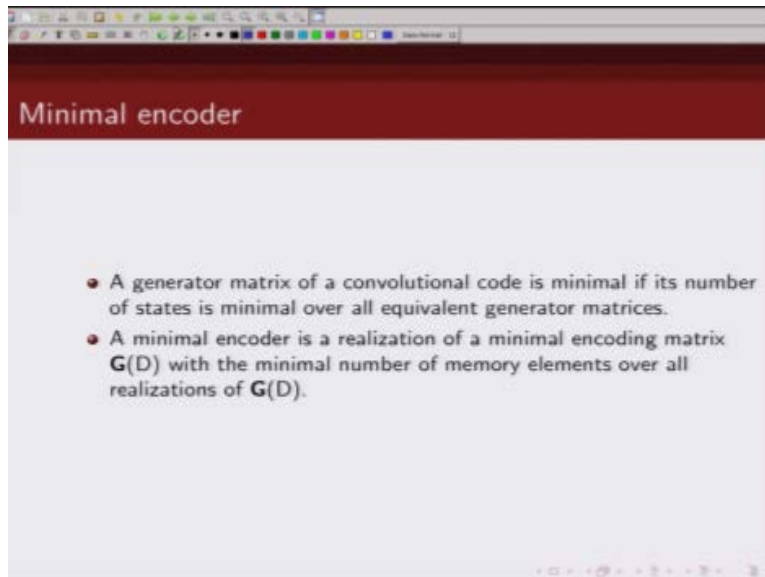


Minimal encoder

- A generator matrix of a convolutional code is minimal if its number of states is minimal over all equivalent generator matrices.
- A minimal encoder is a realization of a minimal encoding matrix  $\mathbf{G}(D)$  with the minimal number of memory elements over all realizations of  $\mathbf{G}(D)$ .

And among the minimal encoder matrix are minimal encoder is basically our realization of a minimal encoding matrix which will result in minimum number of memory elements use to represent that, that particular convolutional encoder. So we define a minimal encoder as.

(Refer Slide Time: 42:21)



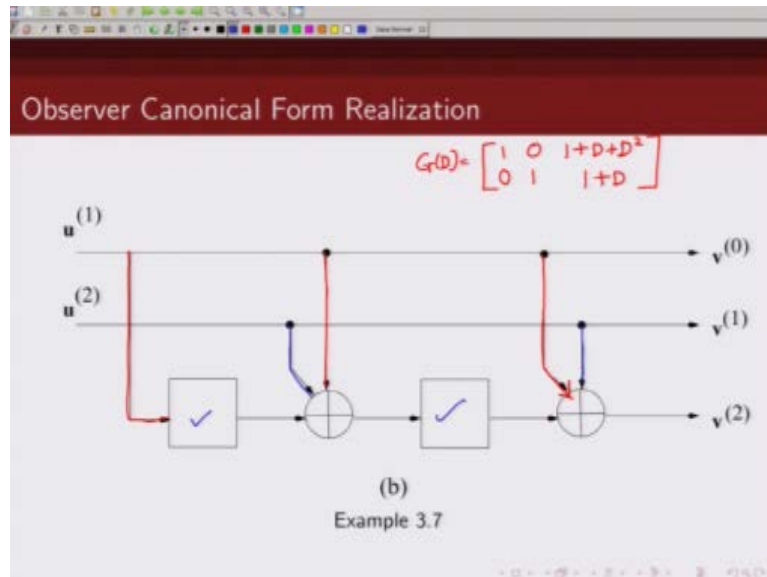
The image shows a presentation slide with a dark red header containing the title "Minimal encoder". Below the header, on a light gray background, are two bullet points. The first bullet point states that a generator matrix of a convolutional code is minimal if its number of states is minimal over all equivalent generator matrices. The second bullet point states that a minimal encoder is a realization of a minimal encoding matrix  $\mathbf{G}(D)$  with the minimal number of memory elements over all realizations of  $\mathbf{G}(D)$ . The slide is framed by a window border with standard OS icons at the top and navigation controls at the bottom.

### Minimal encoder

- A generator matrix of a convolutional code is minimal if its number of states is minimal over all equivalent generator matrices.
- A minimal encoder is a realization of a minimal encoding matrix  $\mathbf{G}(D)$  with the minimal number of memory elements over all realizations of  $\mathbf{G}(D)$ .

The minimal realization of a minimal encoding matrix, so the minimal encoder should result in minimum number of memory elements used to represent that particular convolutional encoder.

(Refer Slide Time: 42:38)



And for the example that we have considered this in this case you can see from the generator matrix the maximum degree of these two so we atleast made two memory elements to represent it and you can see the observer can undergo form realization in this particular example we will result in minimal encoder on this convolutional encoder. So this realization will result in minimal encoder realization for this convolutional encoder. Thank you.

### Acknowledgement

**Ministry of Human Resource & Development**

**Prof. Satyaki Roy**

**Co-ordinator, NPTEL IIT Kanpur**

**NPTEL Team**

**Sanjay Pal**

**Ashish Singh**

**Badal Pradhan**

**Tapobrata Das**

**Ram Chandra**  
**Dilip Tripathi**  
**Manoj Shrivastava**  
**Padam Shukla**  
**Sanjay Mishra**  
**Shubham Rawat**  
**Shikha Gupta**  
**K. K. Mishra**  
**Aradhana Singh**  
**Sweta**  
**Ashutosh Gairola**  
**Dilip Katiyar**  
**Sharwan**  
**Hari Ram**  
**Bhadra Rao**  
**Puneet Kumar Bajpai**  
**Lalty Dutta**  
**Ajay Kanaujia**  
**Shivendra Kumar Tiwari**

**an IIT Kanpur Production**

**©copyright reserved**