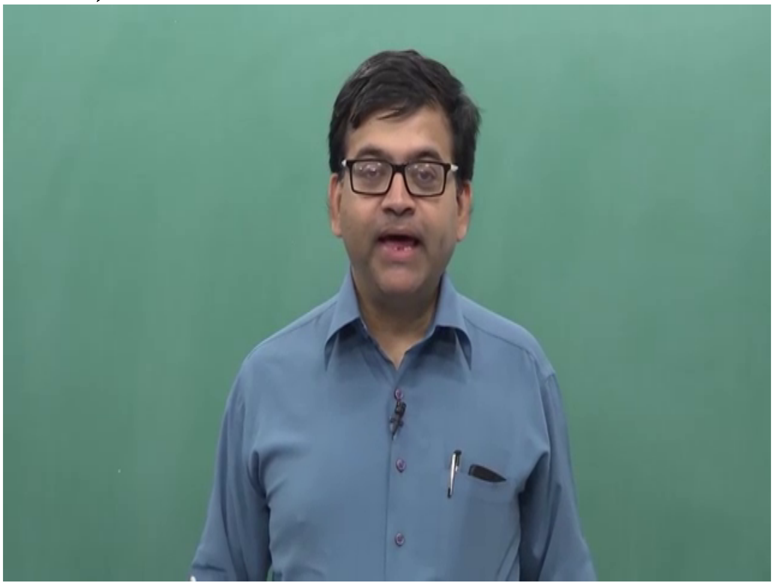
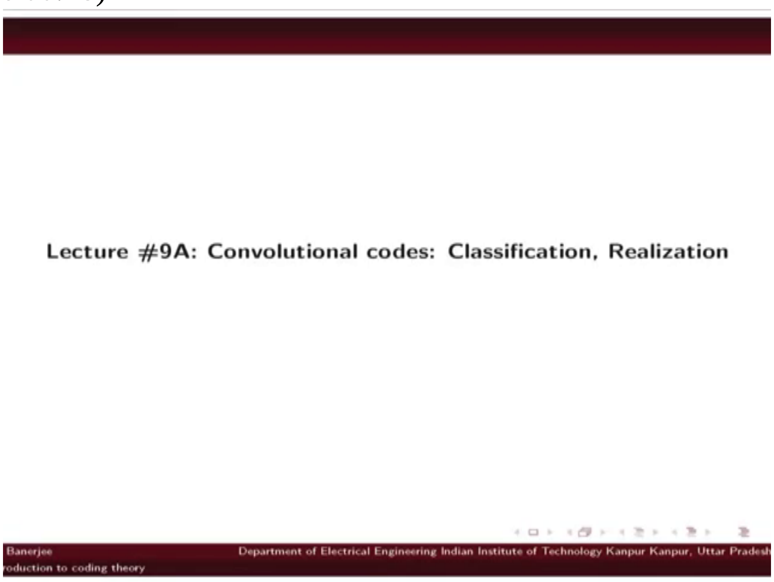
An Introduction to Coding Theory
Professor Adrish Banerji
Department of Electrical Engineering
Indian Institute of Technology, Kanpur
Module 04
Lecture Number 17
Convolutional Codes: Classification, Realization

(Refer Slide Time 00:14)



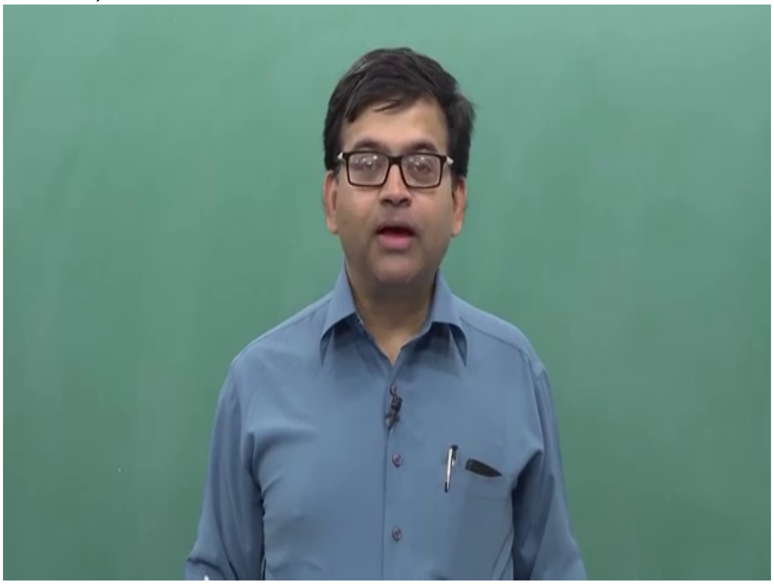
In this lecture today we are going to talk about

(Refer Slide Time 00:18)



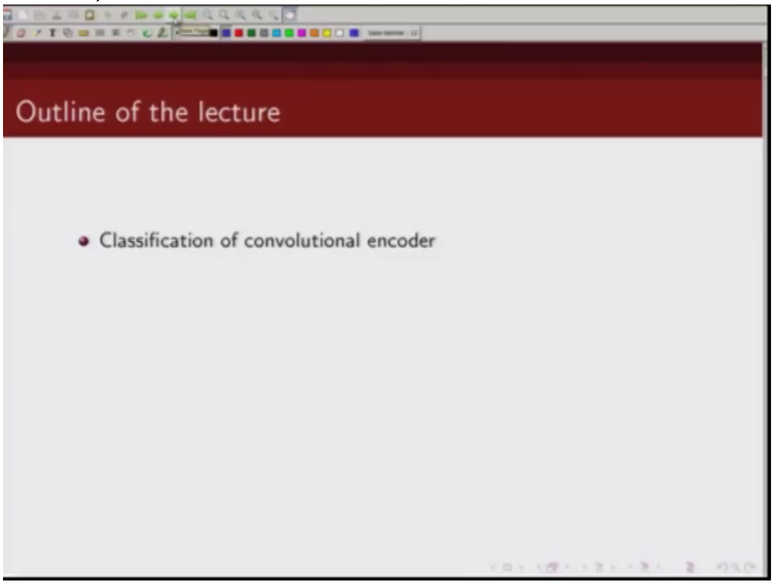
classification of convolutional codes based on

(Refer Slide Time 00:23)



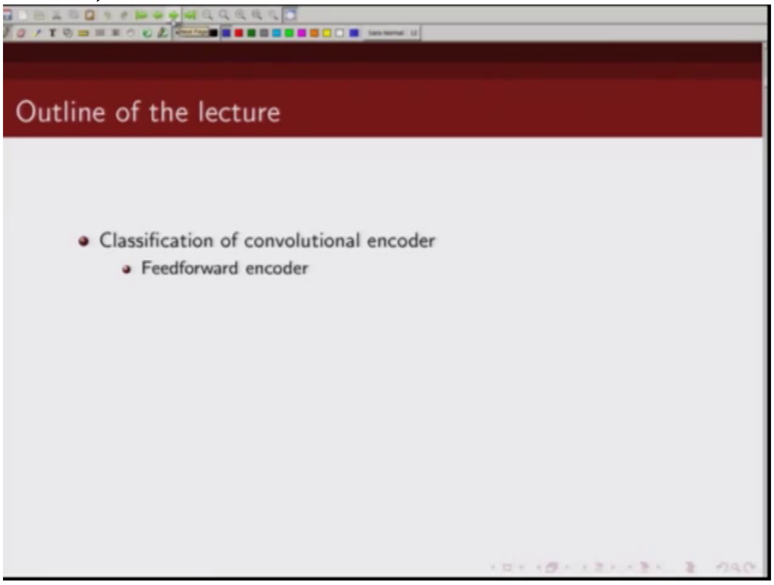
type of connections between the output and the input. Also based on what are our output bits, we will classify convolutional codes into systematic and non systematic codes. Then we are going to talk about how we can realize convolutional codes using shift registers. So

(Refer Slide Time 00:48)



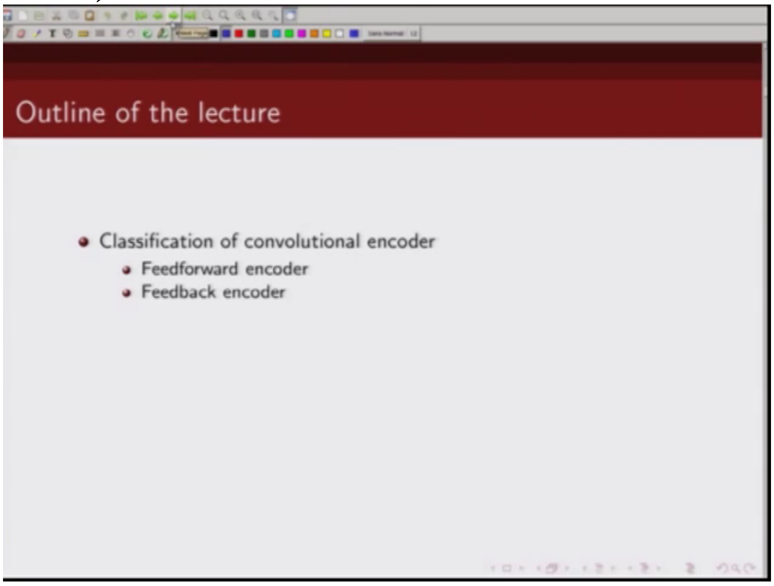
as I said first we will talk about convolutional codes

(Refer Slide Time 00:51)



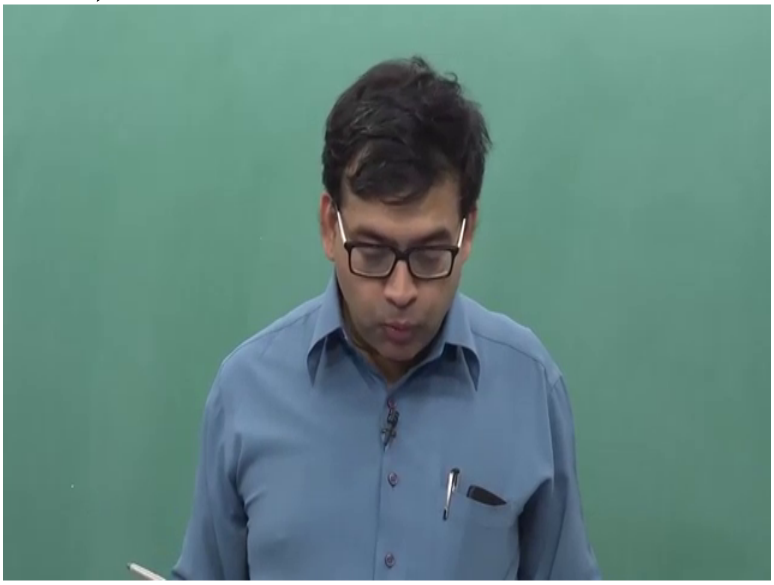
and in this we are going to

(Refer Slide Time 00:55)



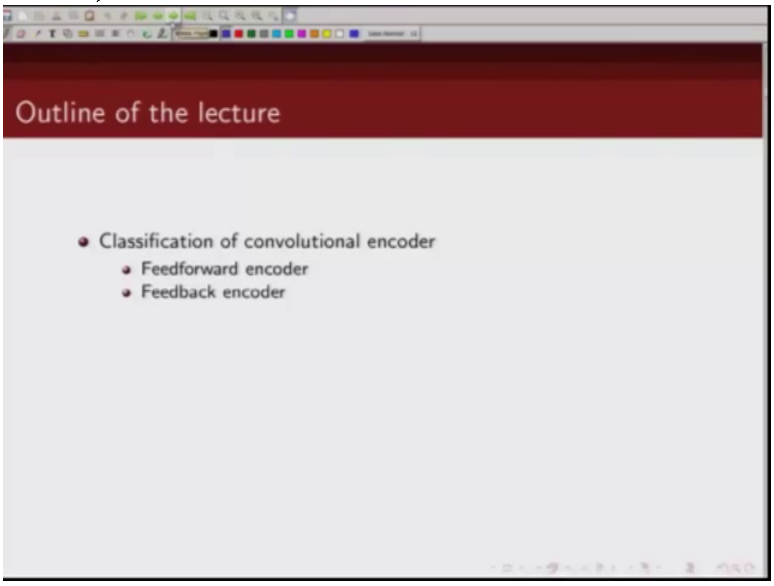
talk about classification based on types of connections between the input and the output of

(Refer Slide Time 01:02)



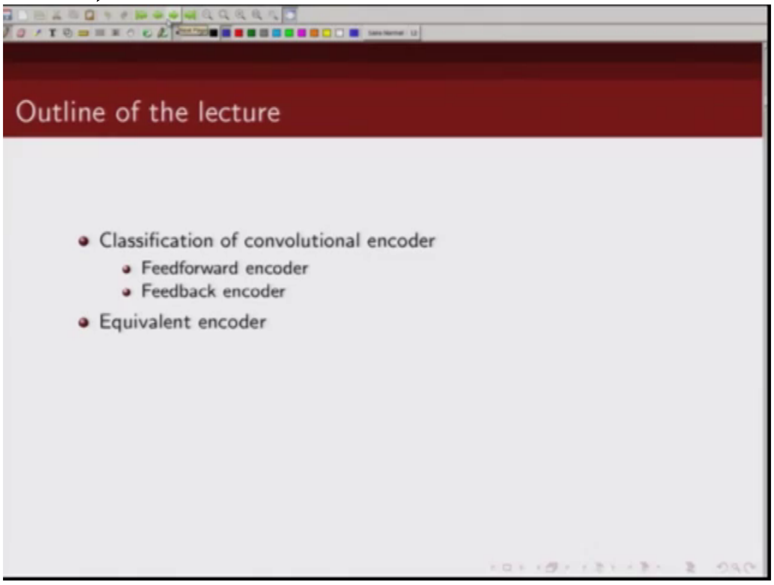
the convolutional encoder. In this regard we are going to talk about what do we mean by feed forward encoder and feedback encoder.

(Refer Slide Time 01:12)



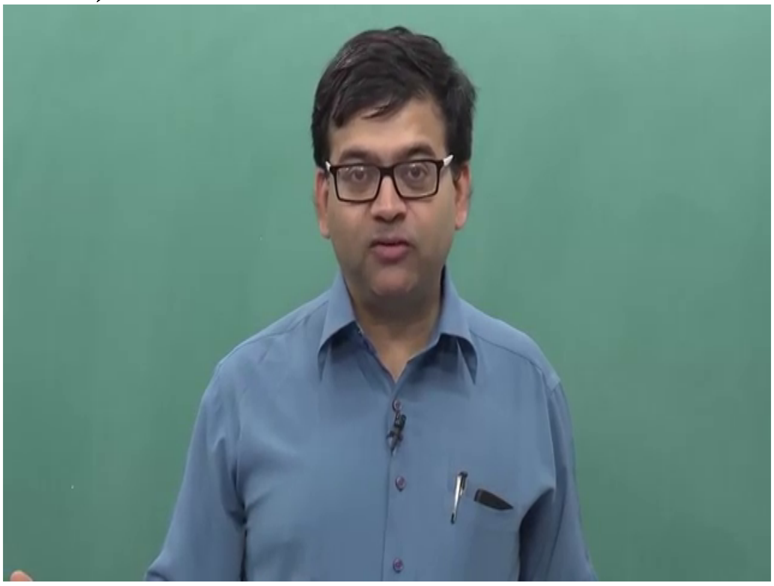
Then we are

(Refer Slide Time 01:13)



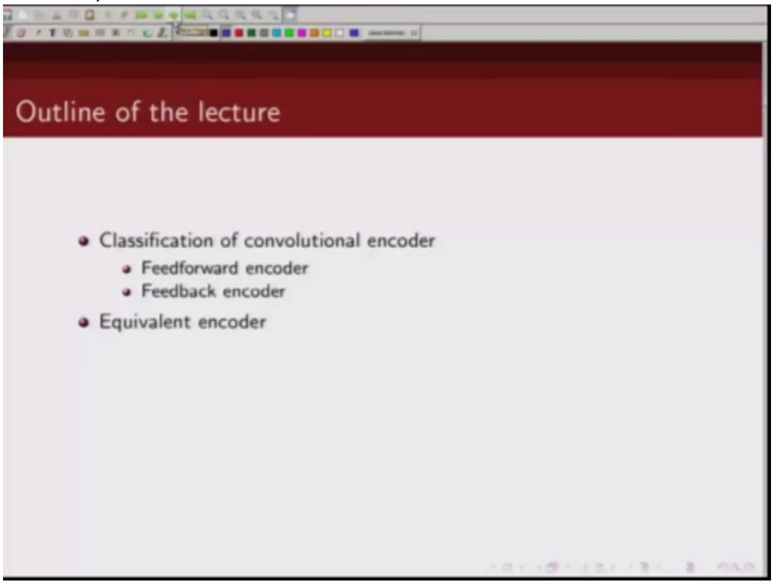
going to introduce a classification based on what are the output bits.

(Refer Slide Time 01:19)



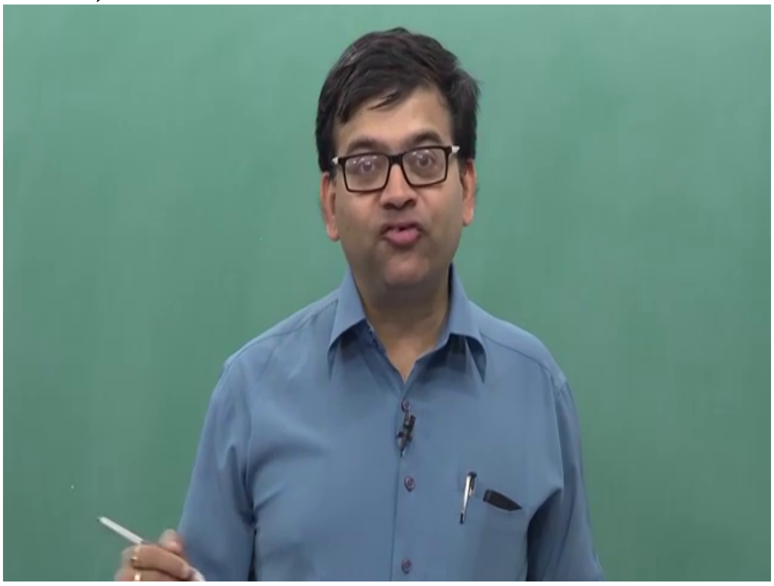
Whether the information bits directly appears in the output or not, based on that there will be a classification of convolutional code,

(Refer Slide Time 01:31)



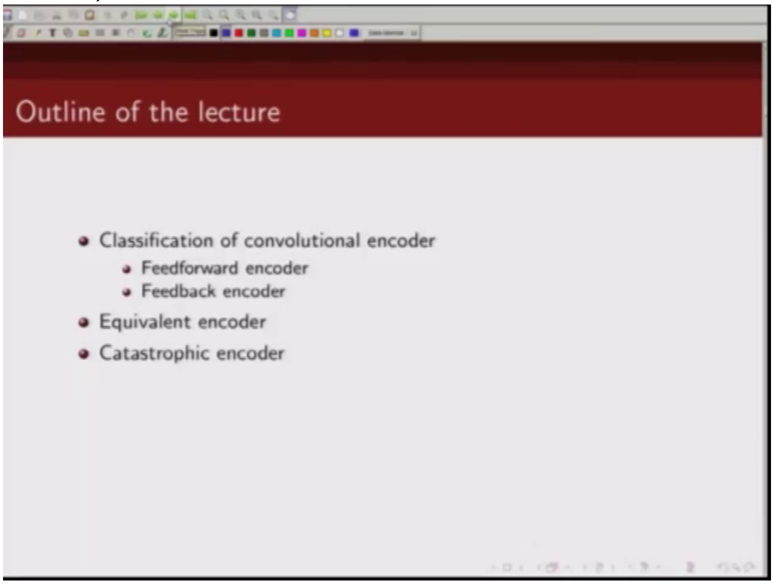
the encoder basically where information bits can be separated out is known as systematic encoder and in

(Refer Slide Time 01:39)



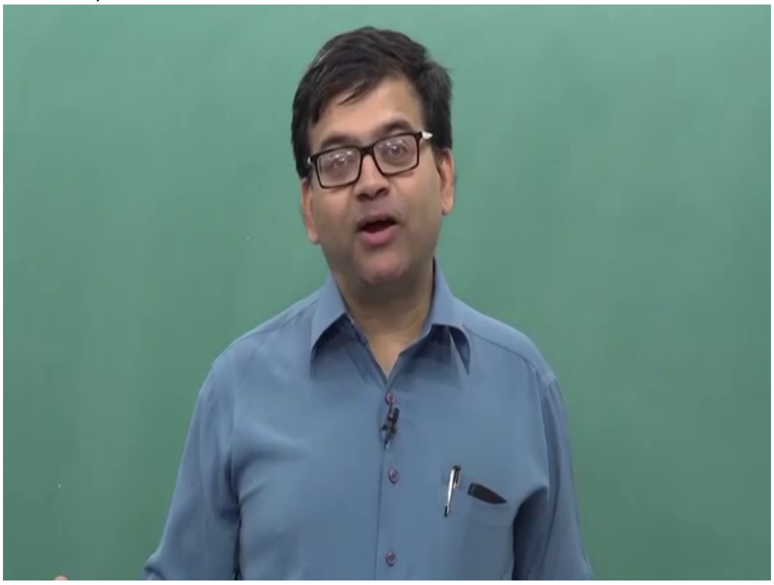
non systematic encoder we cannot separate out information bits directly from the parity bit. So we will talk about what do we mean by systematic encoder for convolutional code and non systematic encoder. And then we will introduce the concept of equivalent encoder.

(Refer Slide Time 02:08)



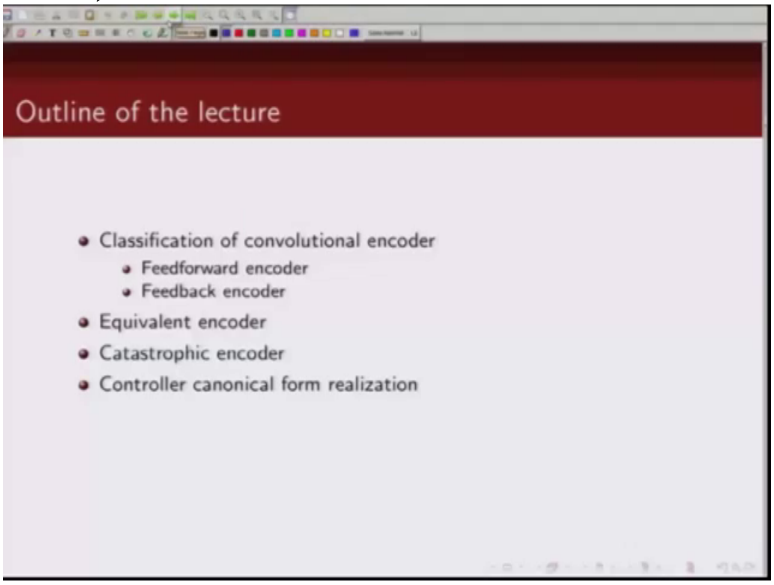
For every non-systematic encoder there is an equivalent systematic encoder and through an example we are going to illustrate how we can get its equivalent encoder. Then we are going to talk about a class of encoder where, if the input bits are very high

(Refer Slide Time 02:18)



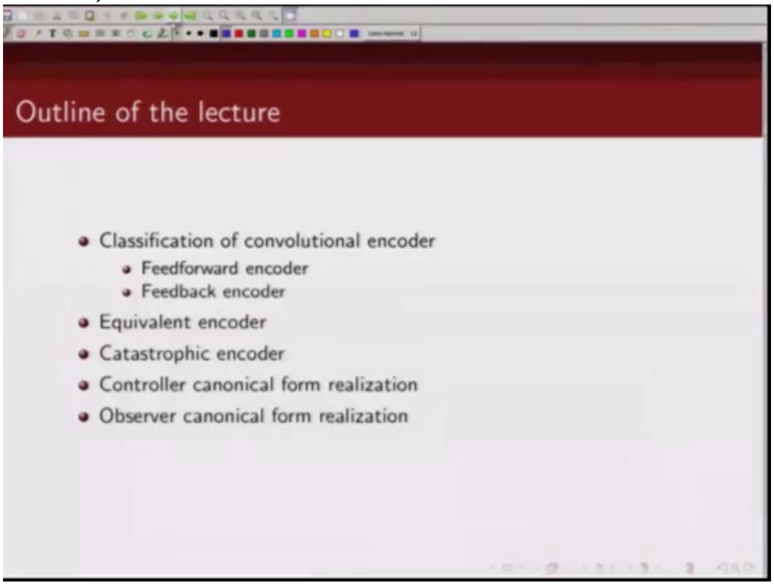
weight we can still get an output codeword of very low weight and these kinds of encoders are known as catastrophic encoders.

(Refer Slide Time 02:29)



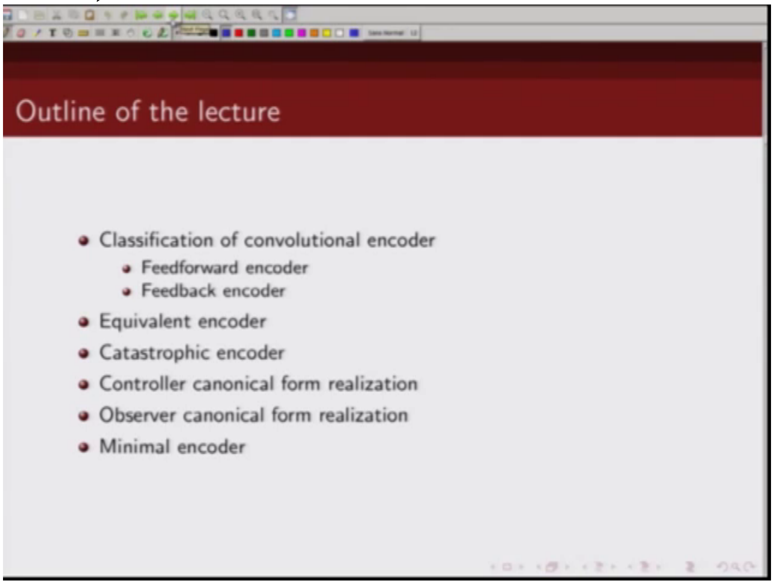
And finally we are going

(Refer Slide Time 02:30)



to talk about 2 different types of realization of convolutional codes using shift registers, the first one which is known as controller canonical form realization and the second one is known as observer canonical form realization. And

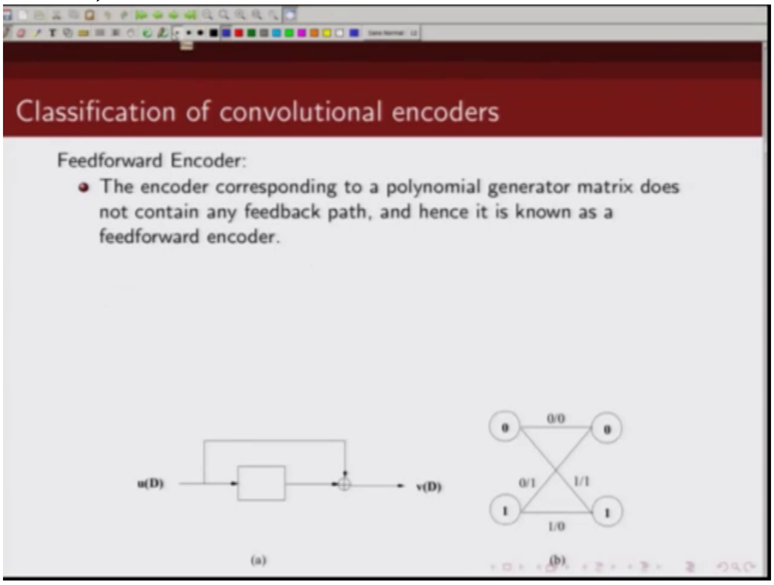
(Refer Slide Time 02:46)



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

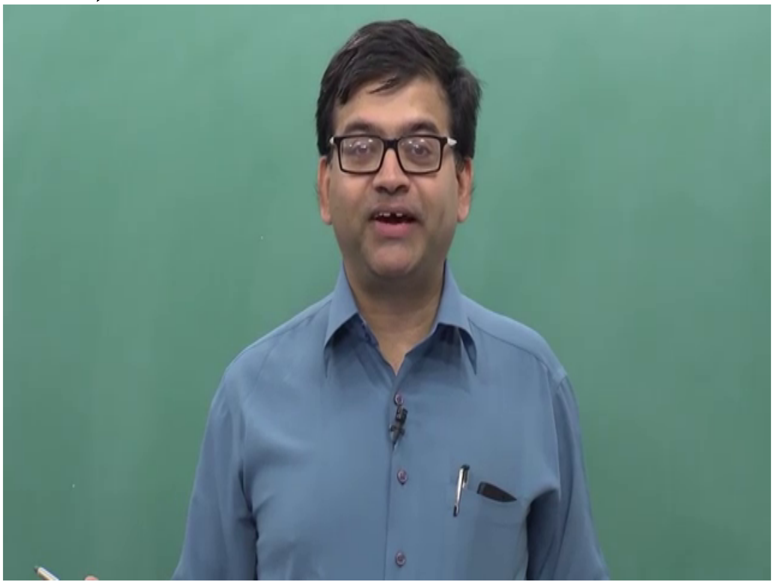
So let us

(Refer Slide Time 02:53)



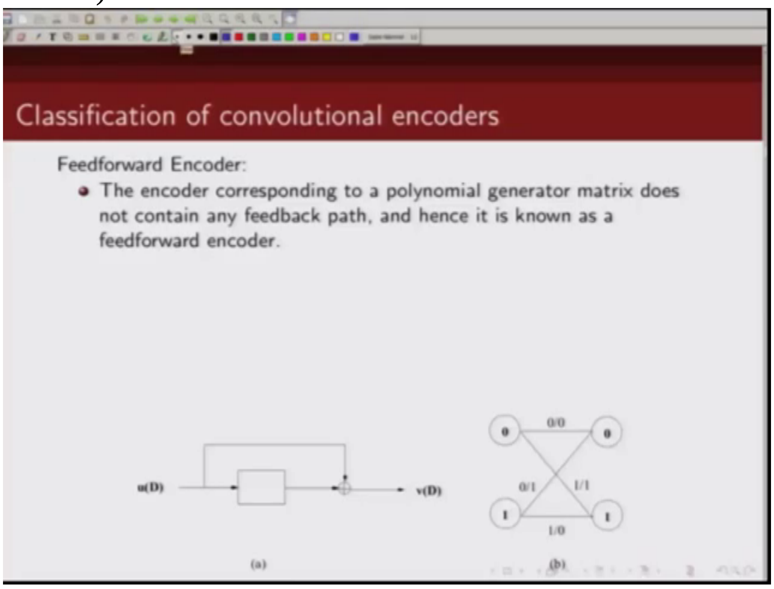
start our discussion on classification of

(Refer Slide Time 02:56)



convolutional encoder. The first type of encoder that we are going to talk about is known as feed forward encoder. So what is a feed forward encoder?

(Refer Slide Time 03:07)

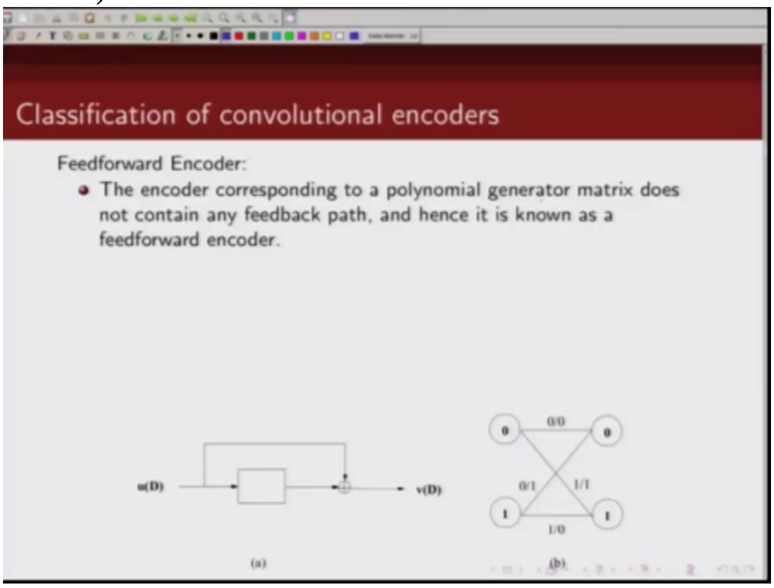


The encoder corresponding to a polynomial generator matrix which does not have any feedback from the output to the input is known as feed forward

(Refer Slide Time 03:19)

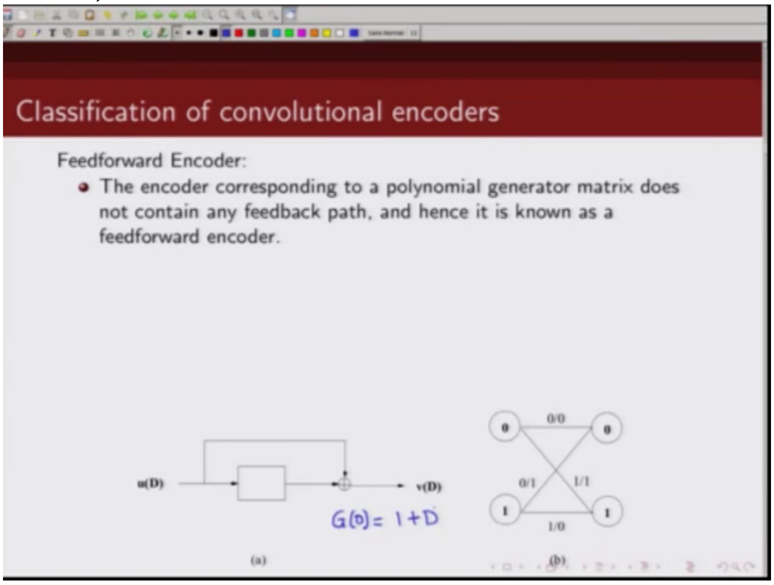
encoder. Let us

(Refer Slide Time 03:21)



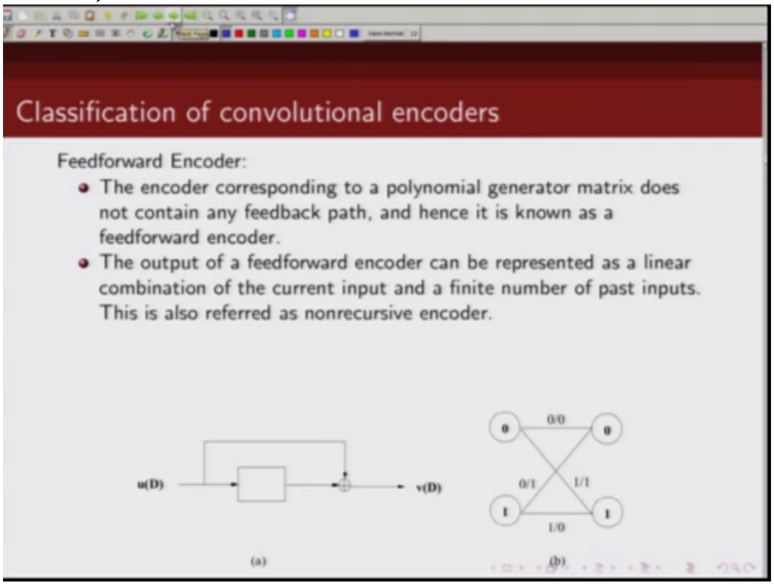
take this example. This is our information sequence. v d denotes our coded sequence. What is the generator matrix G of D, in this case it is given by 1 plus D. Note here

(Refer Slide Time 03:44)



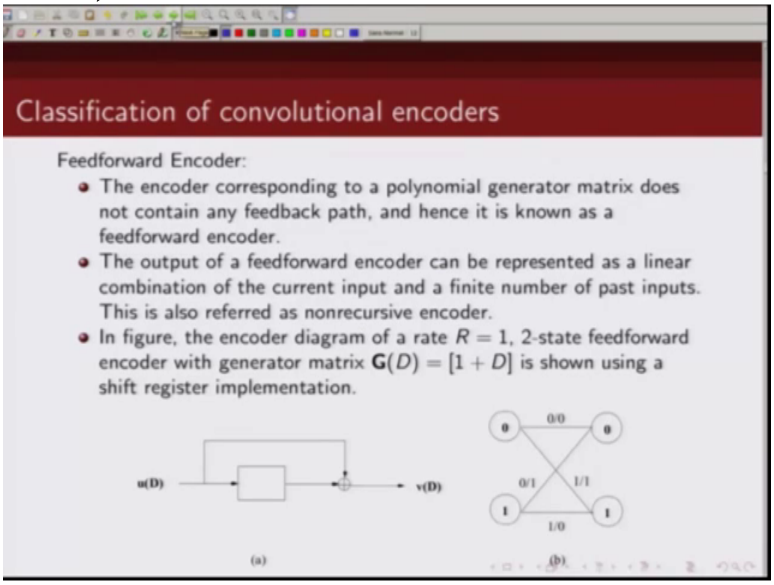
the generator matrix here is a polynomial generator matrix right, as opposed to a rational generator matrix and there is no feedback from the output to the input side. You can see basically the output depends on the current input as well as input one past time instance. So there is no feedback from the output to the encoder side. And this is an example of a feed forward encoder. Now we can

(Refer Slide Time 04:20)



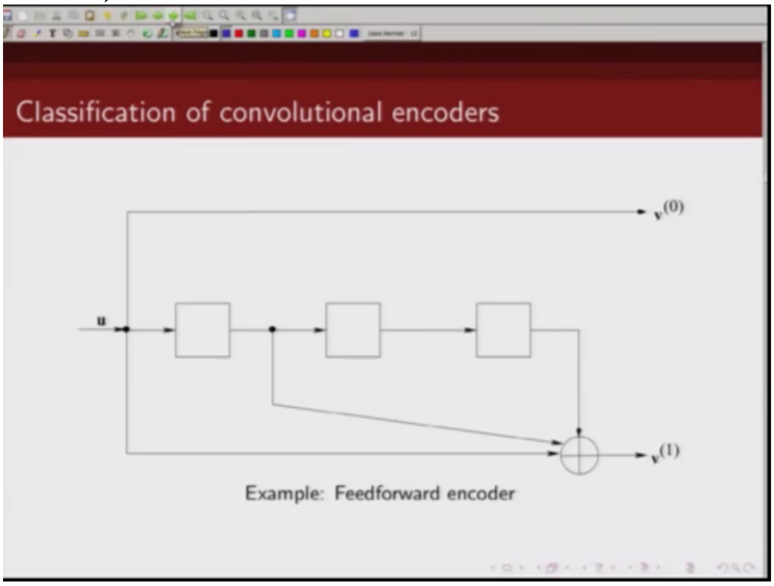
represent the output of feed forward encoder as linear combination of current input and finite number of past inputs. We also refer this type of encoder as non-recursive encoder. And as we said

(Refer Slide Time 04:40)



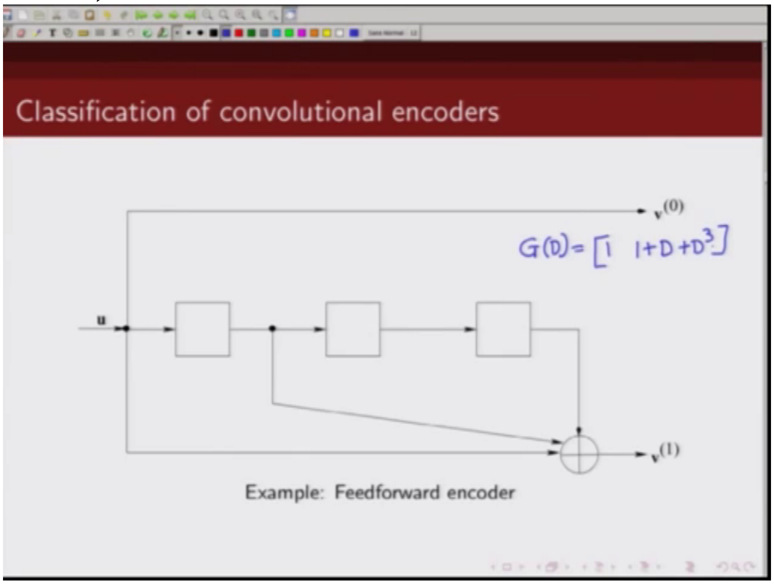
in this we have an example of rate one code, because input one bit, output is one bit coming out and the generator matrix of this rate one code is given by one plus D and you can see this is an example of the feed forward encoder whose generator matrix is a polynomial generator matrix and there is no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

(Refer Slide Time 05:21)



This is another example of feed forward encoder. You can write down the generator matrix of this G of D. v 0 is nothing but 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 cube; this is 1, 2, 3

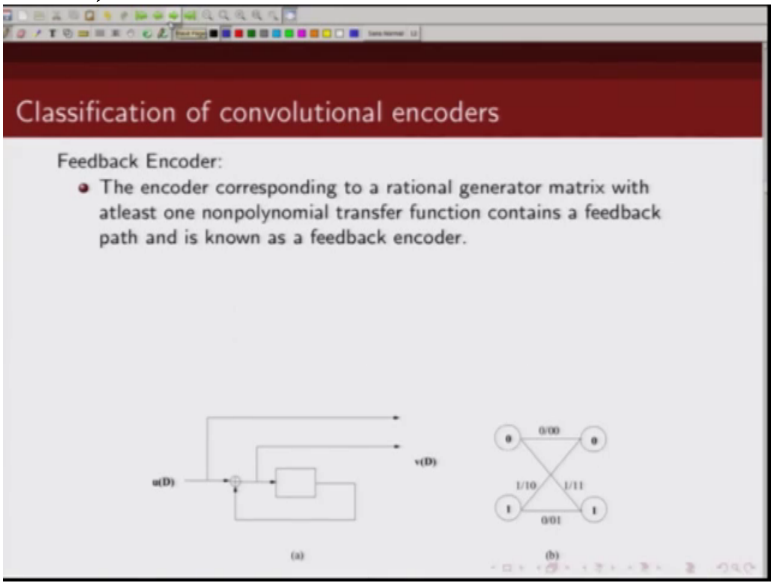
(Refer Slide Time 05:52)



three times instance delayed version of u; so this is a generator matrix, this is also a polynomial generator matrix. There you can see there is no feedback from the output to the input side.

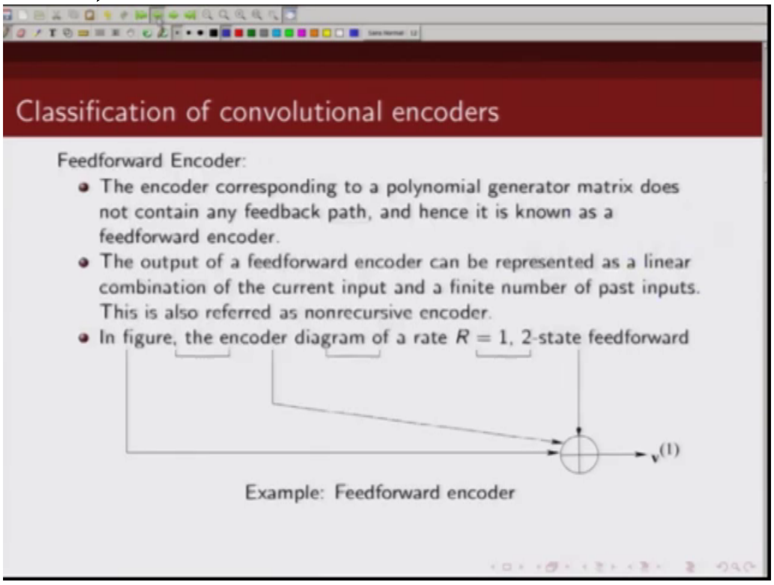
Now let

(Refer Slide Time 06:09)

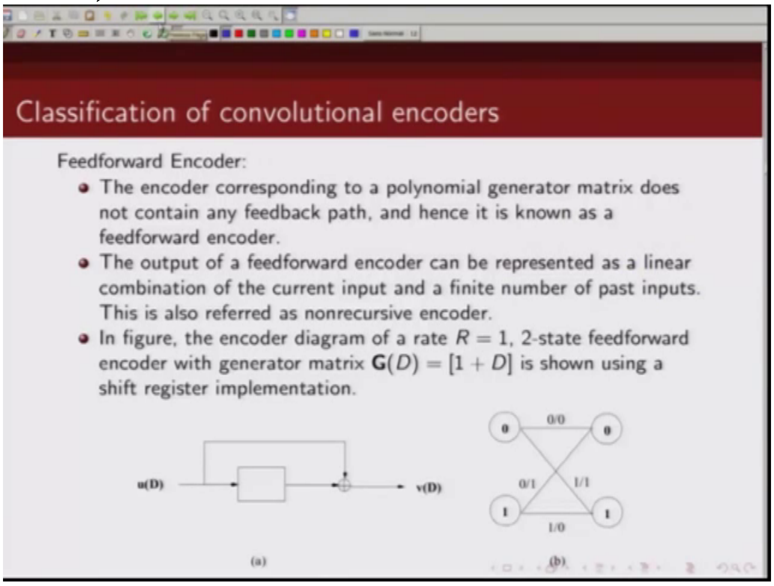


us look at what do we mean by feedback encoder. As opposed to a feed forward encoder, the encoder for a feedback encoder has a rational generator matrix. Please note here we had a

(Refer Slide Time 06:29)

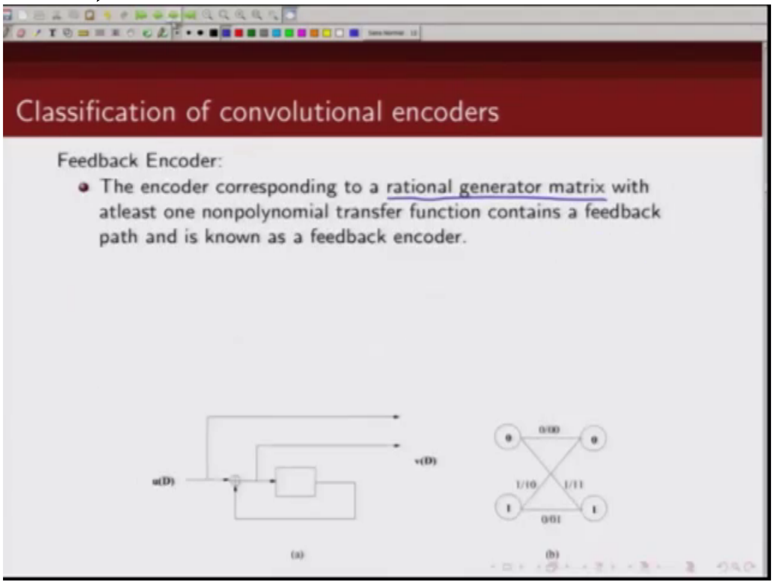


(Refer Slide Time 06:30)



polynomial generator matrix; for a feed forward encoder we had a polynomial generator matrix where as for a

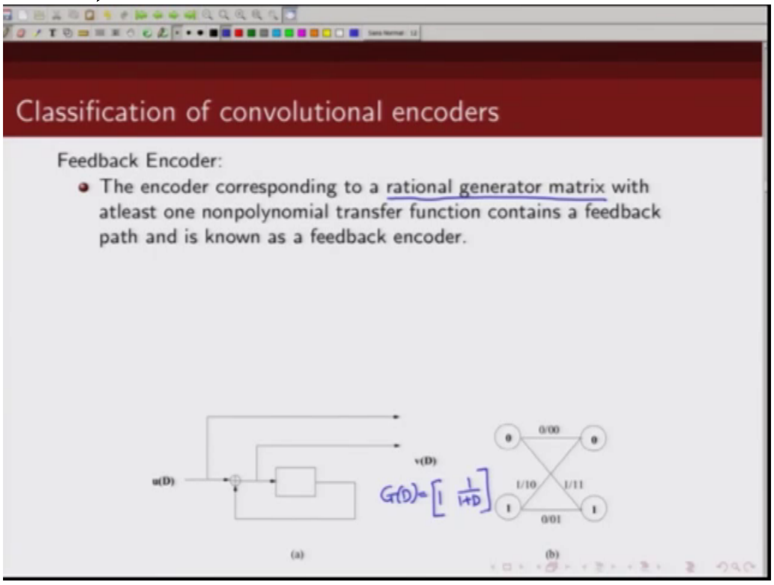
(Refer Slide Time 06:39)



feedback encoder we have a rational generator matrix with at least one non polynomial transfer function containing a feedback path from the output to the input.

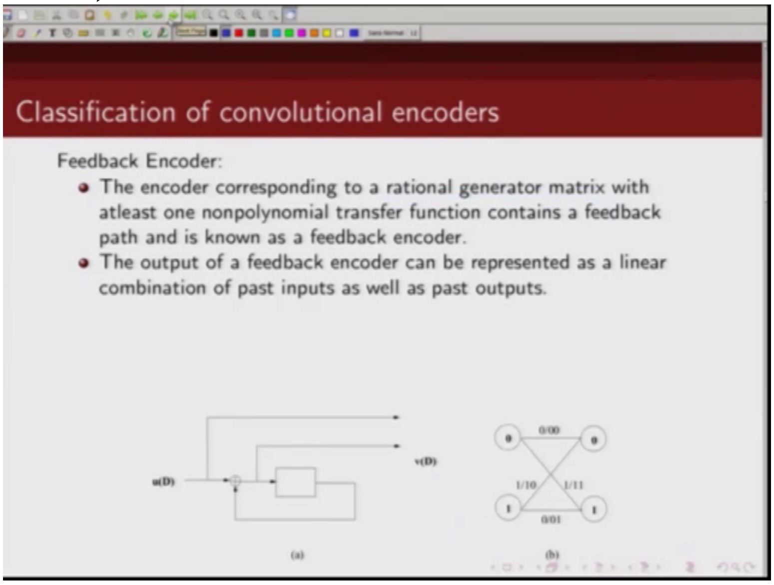
Look at this example. From the output we can see the feedback going into the input side. And the generator matrix for this is basically, so first coded bit is nothing but information bit, so that's 1. And this is basically 1, 1 plus D.

(Refer Slide Time 07:16)



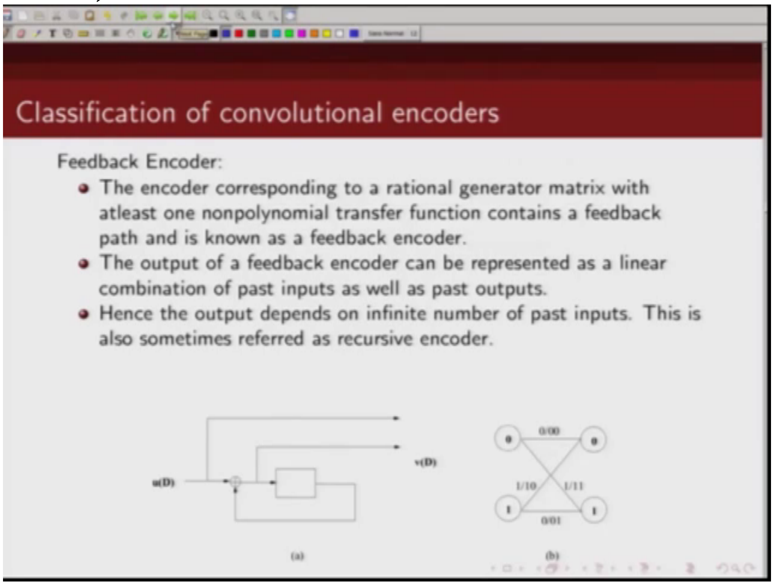
So

(Refer Slide Time 07:18)



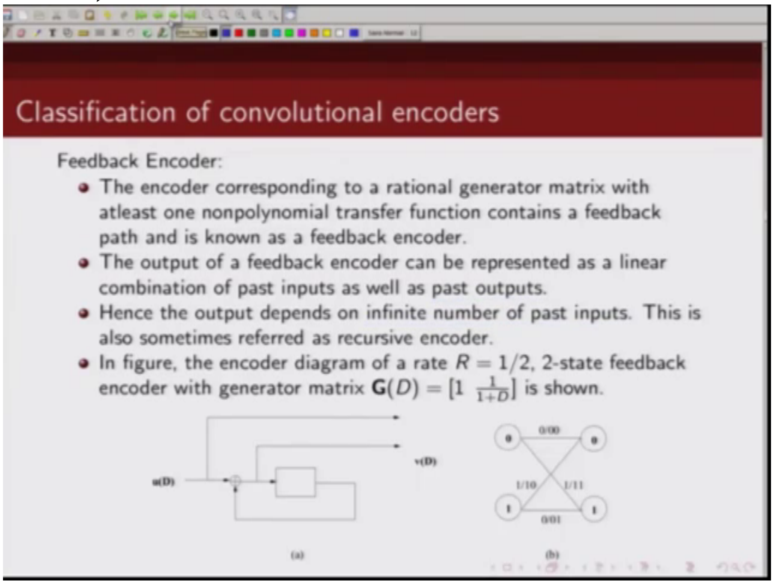
because there is a feedback from the output to the input side, the output of the feedback encoder can be written as a combination of past input as well as past output.

(Refer Slide Time 07:35)



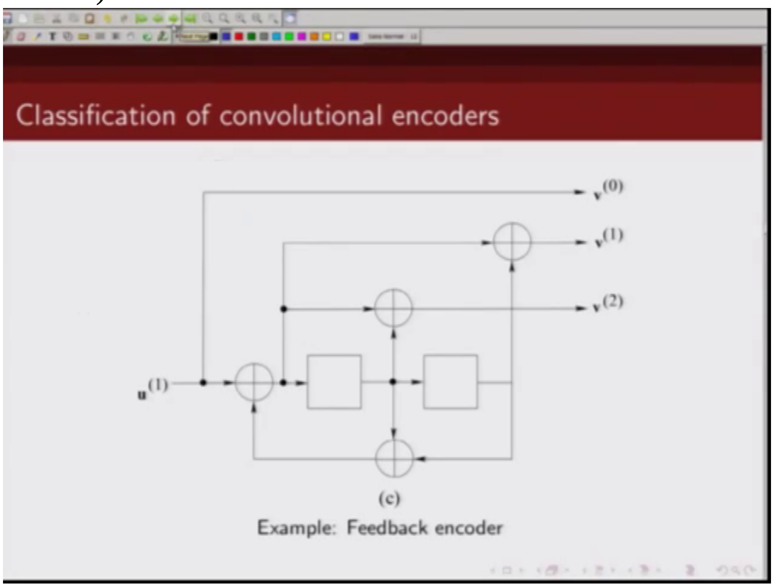
Hence the output depends on infinite number of past input because the current output depends also on past output and past output also depends on past input and past output. So the output will basically depend upon infinite number of past inputs. Now feedback encoder is also known as recursive encoder. And we just now mentioned one example

(Refer Slide Time 08:09)



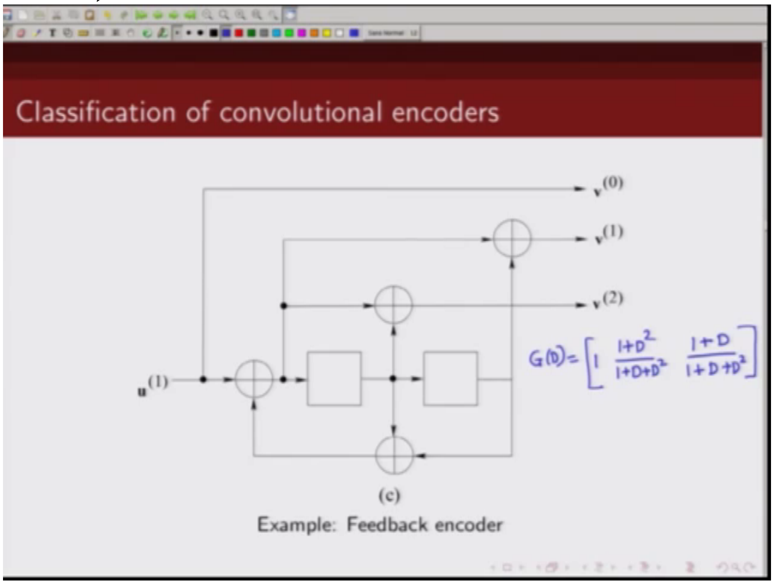
of this feedback encoder is given in this (()). This is a rate one half code. You can see for 1 input we have 2 outputs and its generator matrix is given by this. This is its corresponding state diagram for this feedback encoder. This is another example

(Refer Slide Time 08:34)



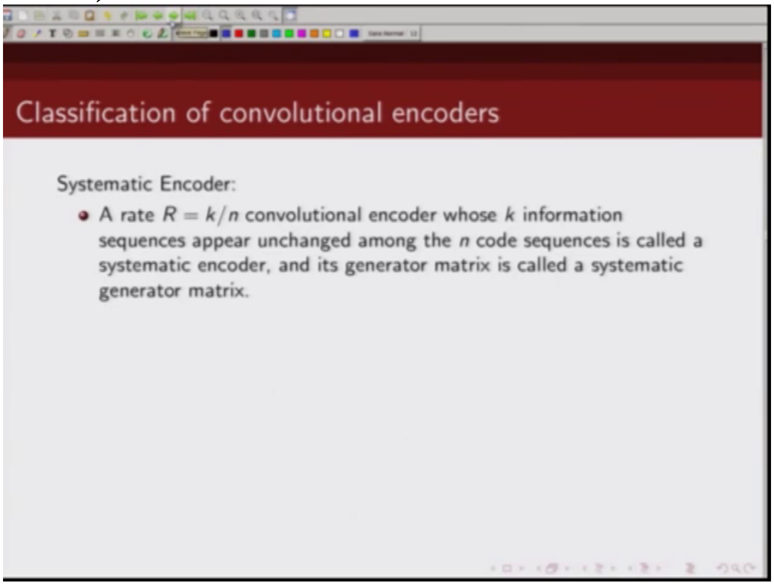
of a feedback encoder. So there is 1 input and there are 3 outputs. We can write the generator matrix G D, the first output is nothing but information bits that's 1. Now what are the feed forward terms in v 1? So v 1 depends on this bit and this bit. So this is 1 plus D square and what's the denominator term, we have basically 1 plus D plus D square term. Similarly v 2 is basically given by 1 plus D and this is 1 plus D plus D square. So this is a generator matrix

(Refer Slide Time 09:31)



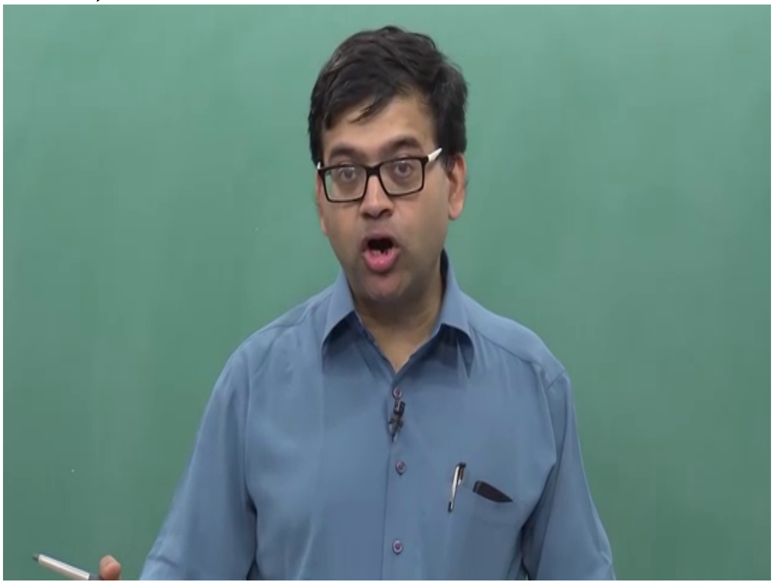
for this feedback encoder.

(Refer Slide Time 09:37)



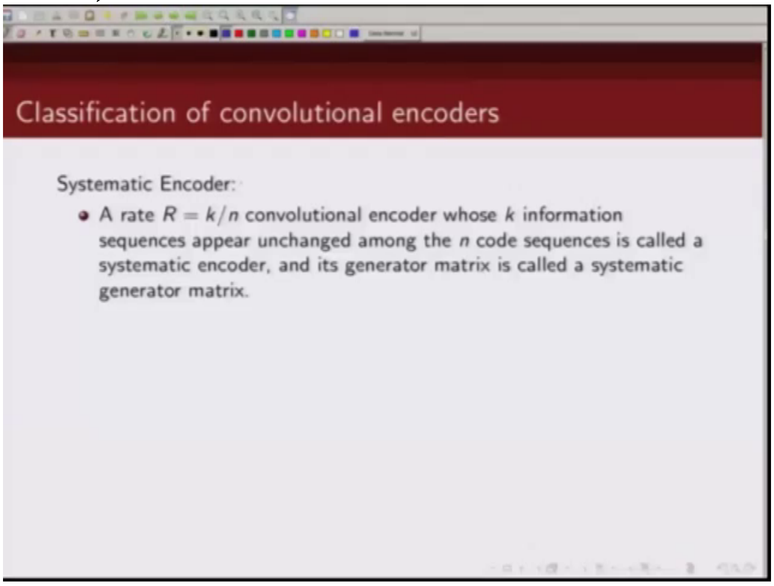
The next classification that we are going to talk about is based

(Refer Slide Time 09:42)



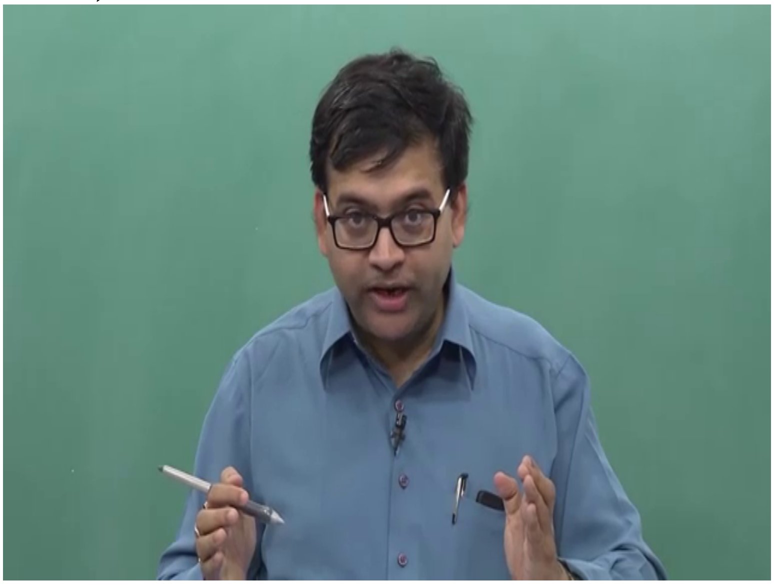
on output bits, whether we can separate out information bits from the coded bits. So in a systematic encoder;

(Refer Slide Time 09:52)



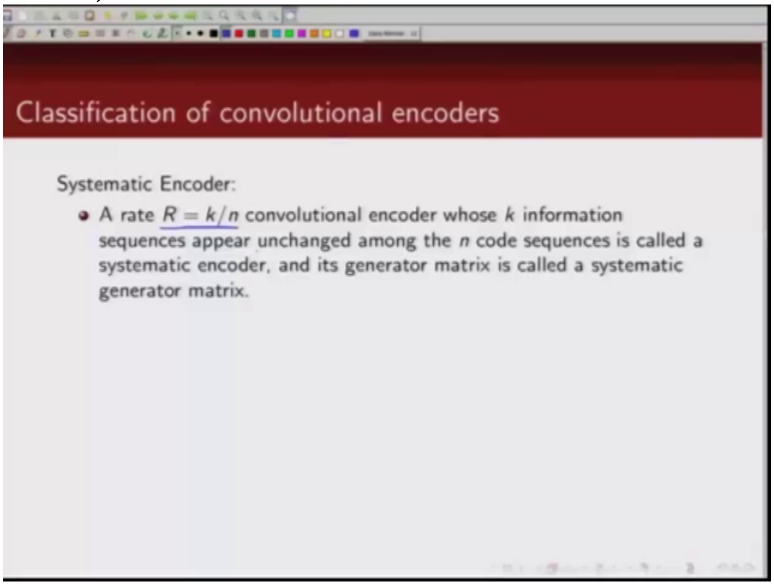
a rate k by n systematic encoder the k information bits appear unchanged in the output. So out of those n coded bits

(Refer Slide Time 10:06)

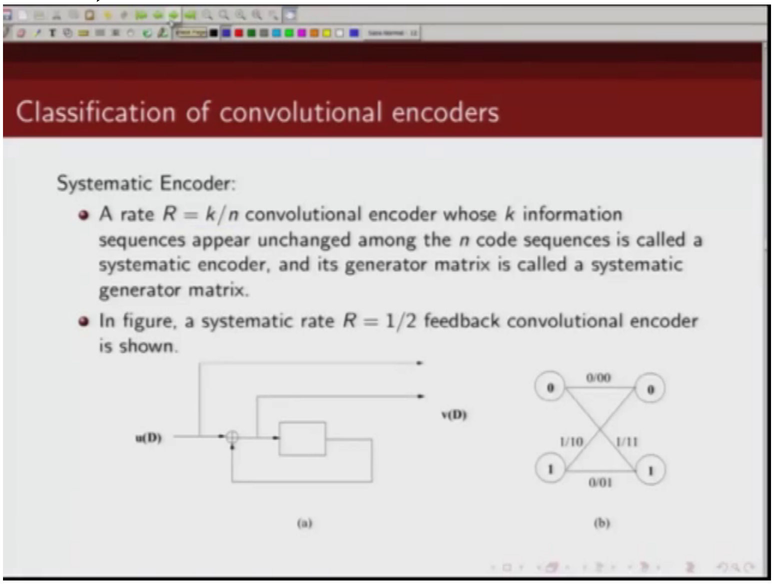


you can directly see the k information bits and rest n minus k bits are your parity bits.

(Refer Slide Time 10:17)



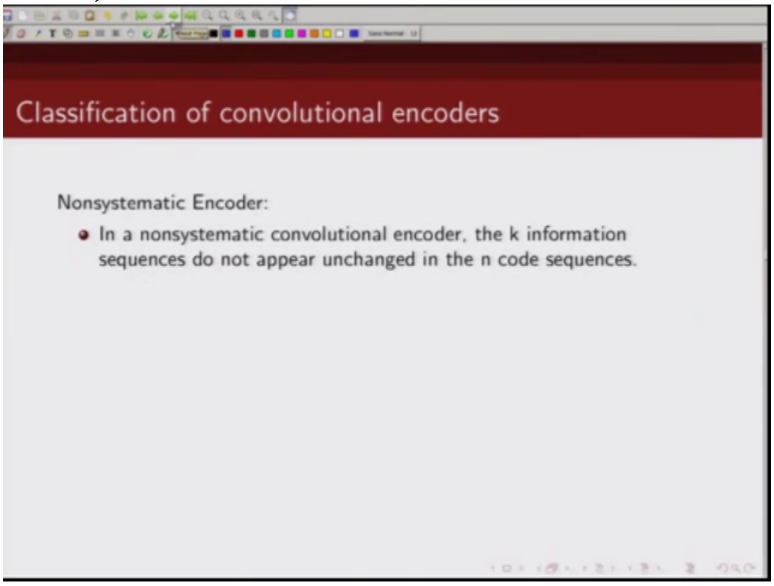
(Refer Slide Time 10:19)



And the generator matrix corresponding to a systematic encoder is known as systematic generator matrix. Take example of this rate one half feedback encoder. You can see there is one input and there are two outputs. So it's rate one half. 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 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.

As opposed

(Refer Slide Time 11:24)

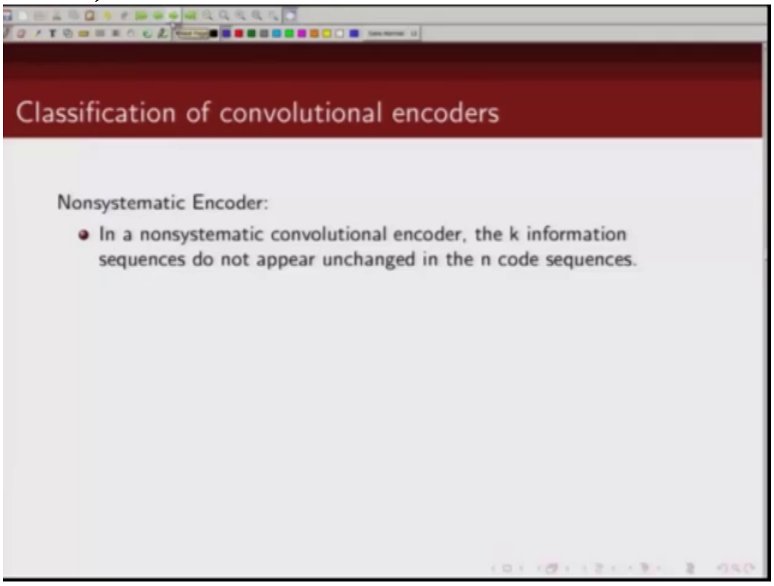


to a systematic encoder, in

(Refer Slide Time 11:26)

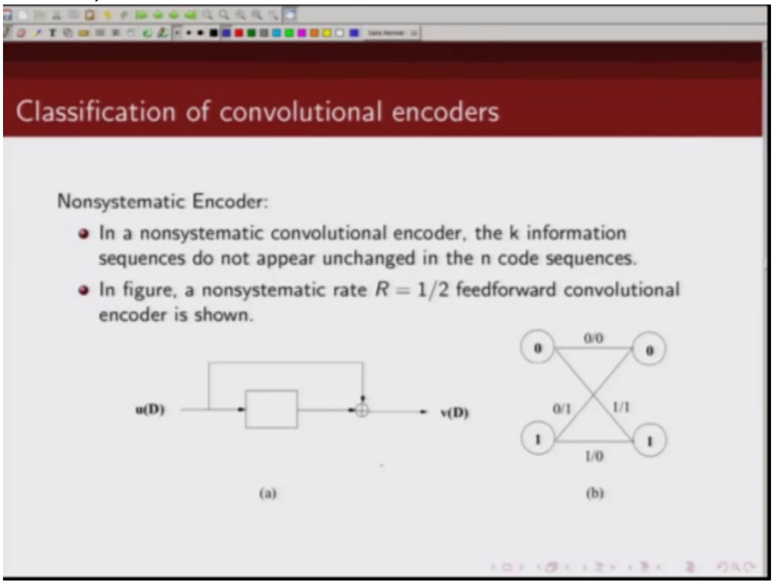
a non-systematic encoder

(Refer Slide Time 11:30)



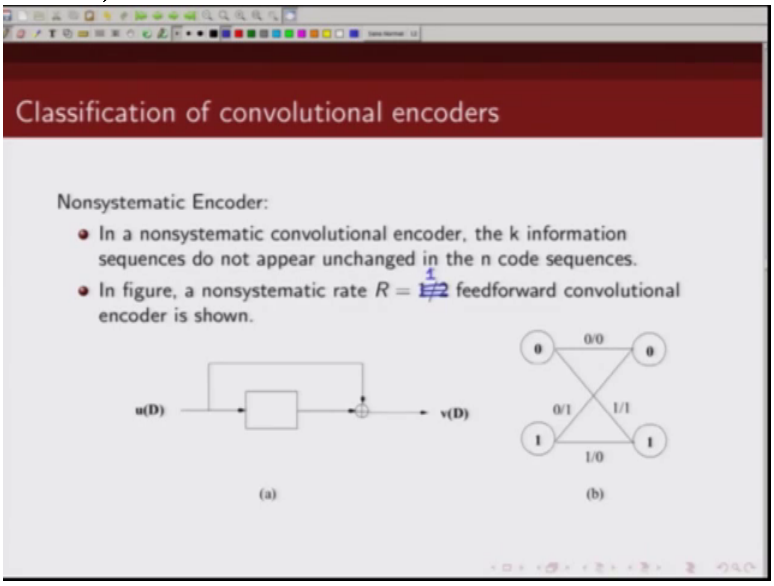
we cannot separate out the k information bits from the n coded bits. This is an example

(Refer Slide Time 11:38)



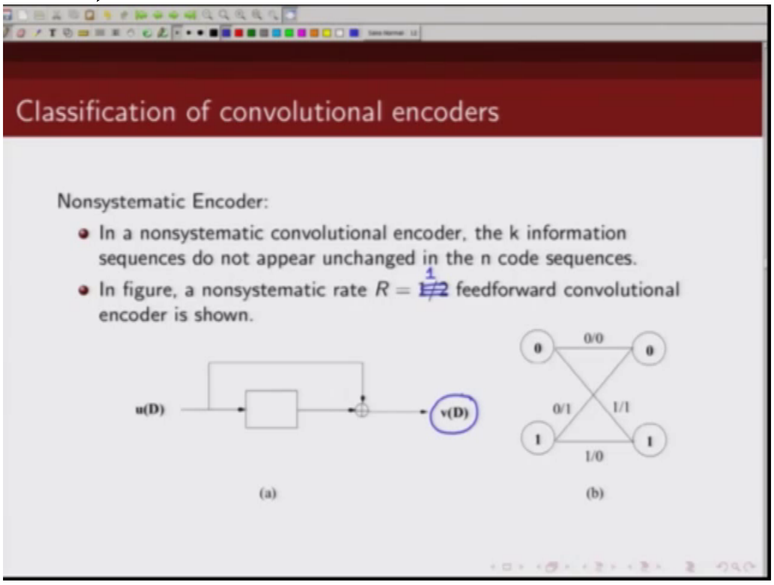
of a, one second I want to make it rate one, this is actually rate, this is a typo, this is rate 1 code

(Refer Slide Time 11:48)



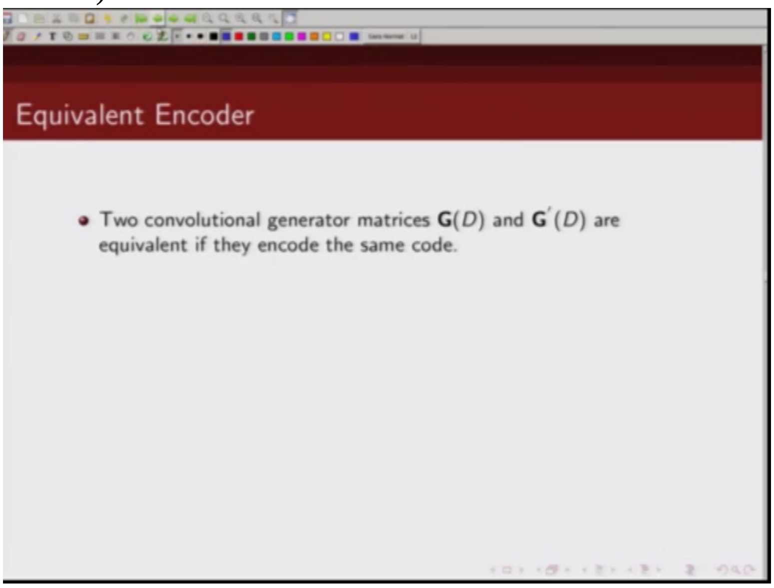
because there is 1 input and there is 1 output this is the rate 1 and this is the feedback, feed forward encoder you can see there is no feedback from the output to the input side; so it is rate 1 feed forward encoder and you can see the output bit

(Refer Slide Time 12:10)



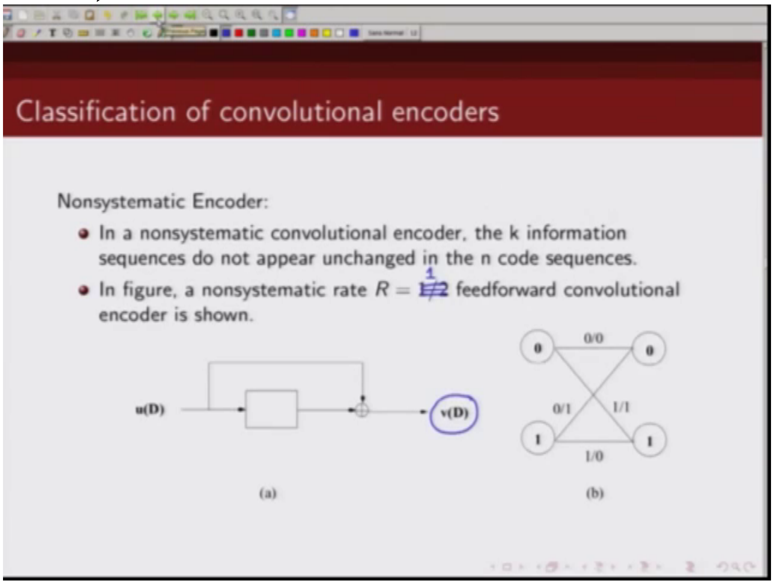
is given by this current input bit and this past input bit. So you cannot directly take out the information bits from this coded bit. So this is an example of a non systematic encoder. We could also define

(Refer Slide Time 12:34)



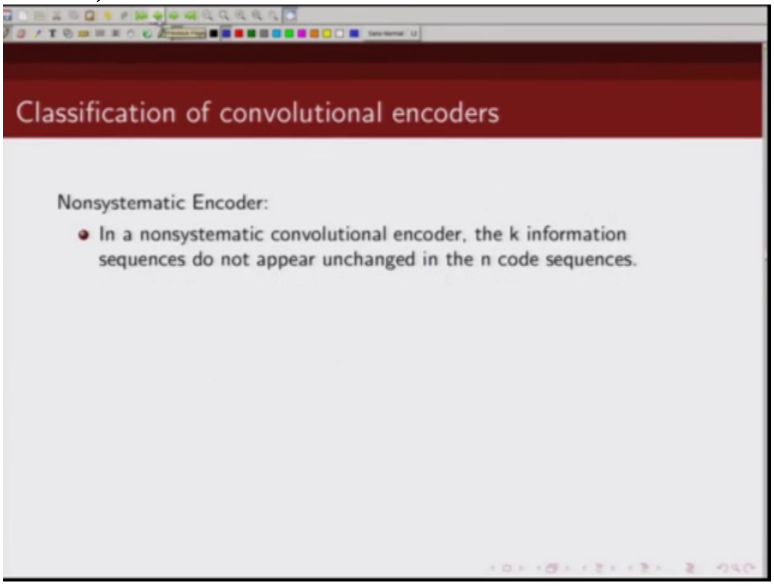
a class

(Refer Slide Time 12:35)



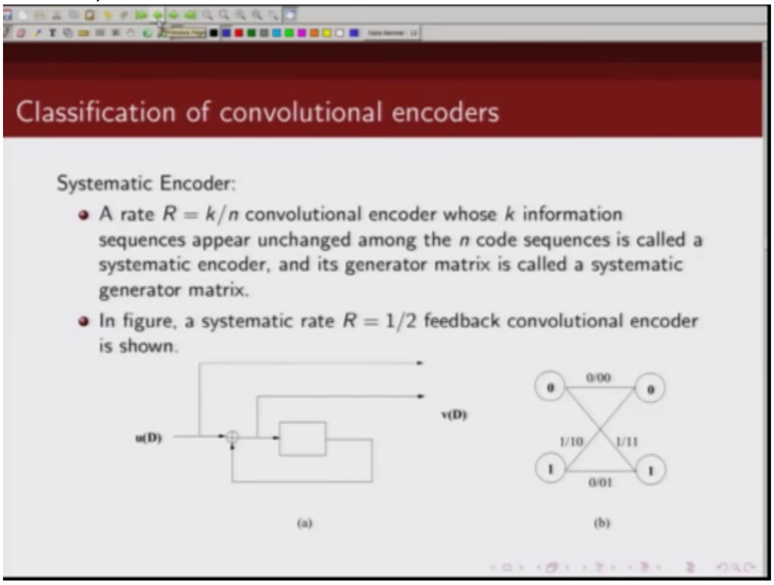
which is called

(Refer Slide Time 12:37)



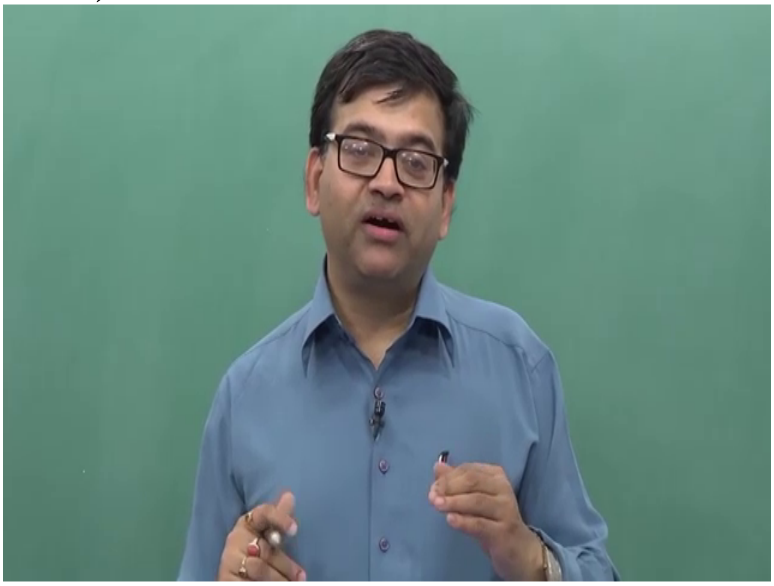
partially systematic

(Refer Slide Time 12:38)



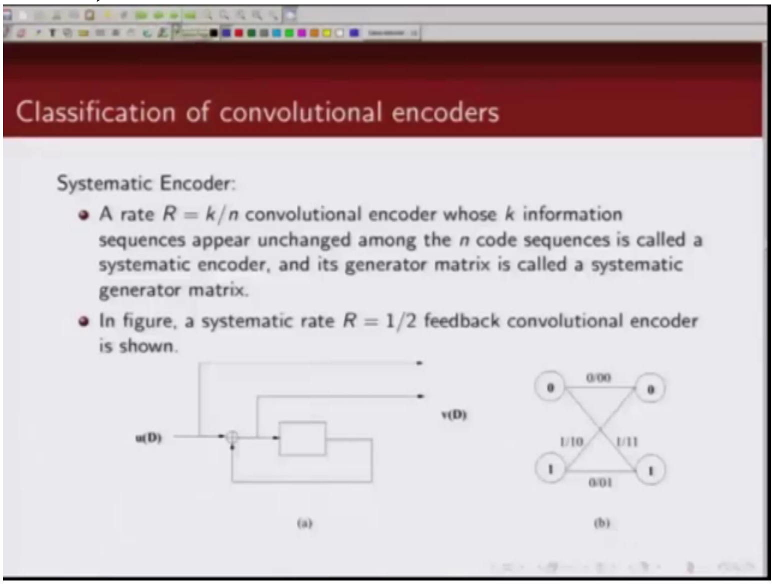
encoder. In a partially systematic encoder,

(Refer Slide Time 12:43)



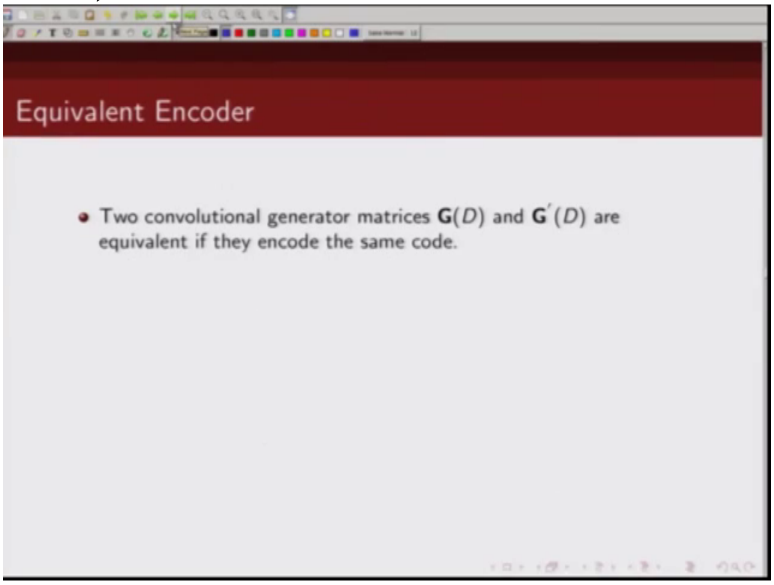
so if you have a rate k by n partially systematic encoder, out of those k information bits some of them appear unchanged in the output while some of the information bits do not appear unchanged in the coded bit. So in a systematic rate k by n encoder we can see directly the k information bits. In a partially systematic encoder we can see a fraction of these k information bits, may be a few bits like 1 to k minus 1 and in a non systematic encoder we cannot see any systematic bit direct, any information bit directly in the output. So all

(Refer Slide Time 13:38)



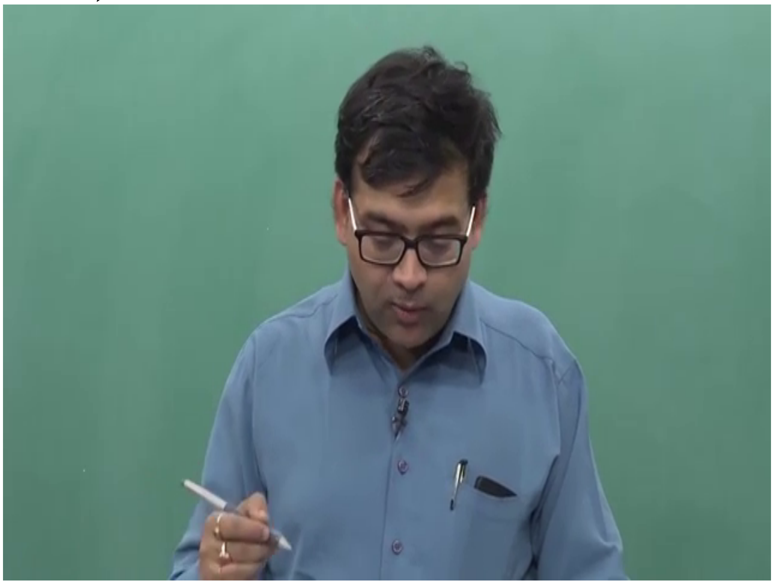
the parity bits are essentially linear combinations of current and past inputs and outputs. Now

(Refer Slide Time 13:48)



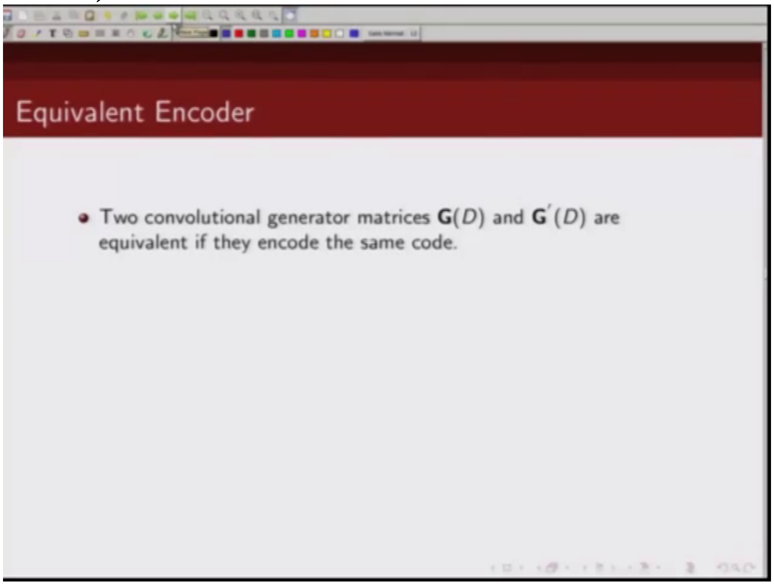
that brings us to our next

(Refer Slide Time 13:52)



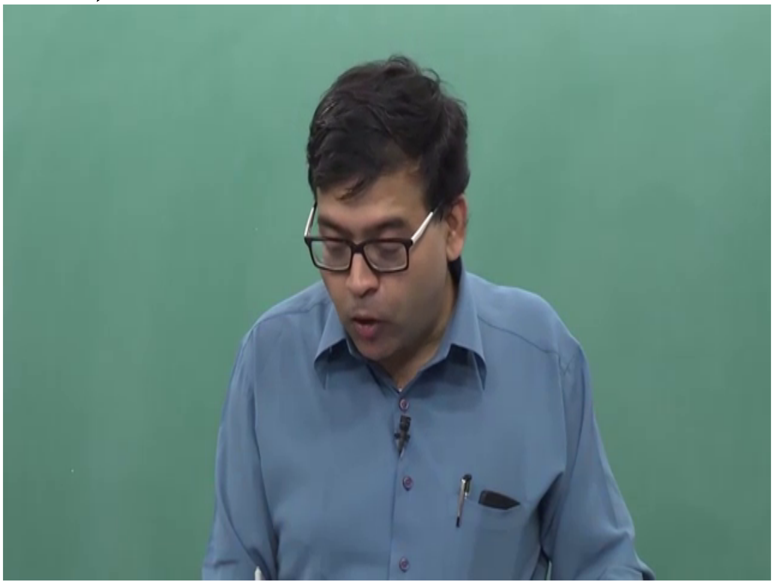
topic of discussion which is concept of equivalent encoders. So before we define

(Refer Slide Time 13:58)



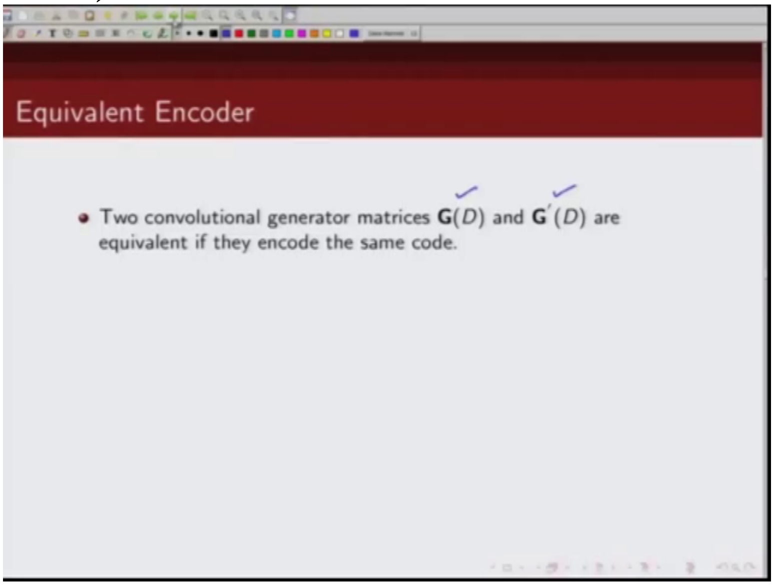
what is an equivalent encoder, we will define what do we mean by equivalent generator matrix. So we, two convolutional matrix let us call it G D and G prime D are equivalent if they encode the same code. Now what do we mean by encode the same code? So the set of codewords generated by this and this, if they are same, then these generator matrix are equivalent. Now the set of codewords 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

(Refer Slide Time 14:49)



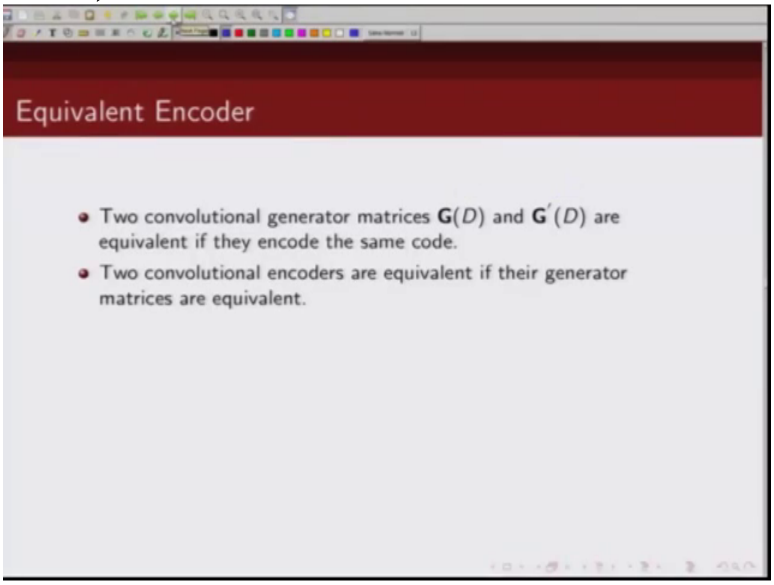
is for this generator matrix.

(Refer Slide Time 14:55)



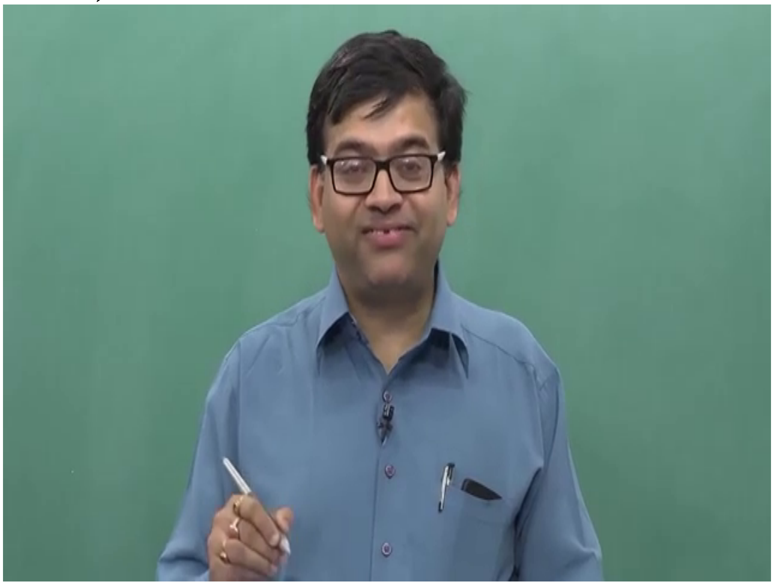
Now we say

(Refer Slide Time 14:56)



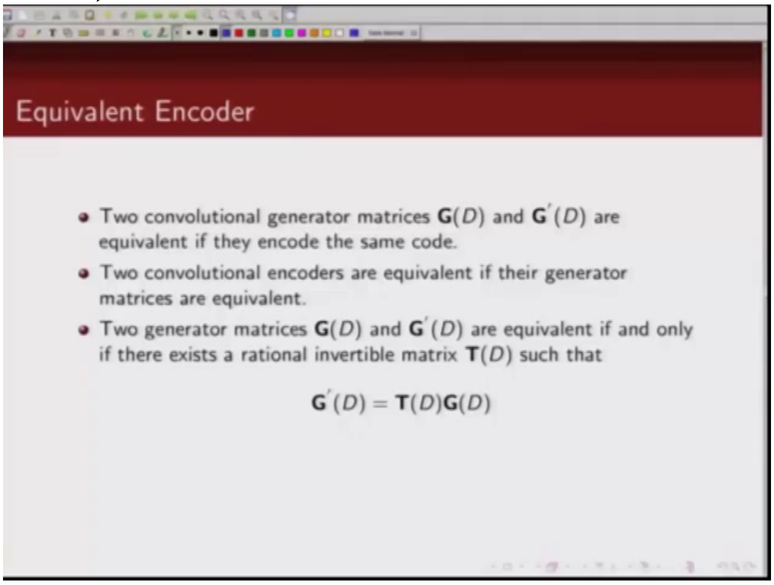
two convolutional encoders are equivalent if their generator matrix are also equivalent. In other words if their generator matrix encode the same

(Refer Slide Time 15:08)



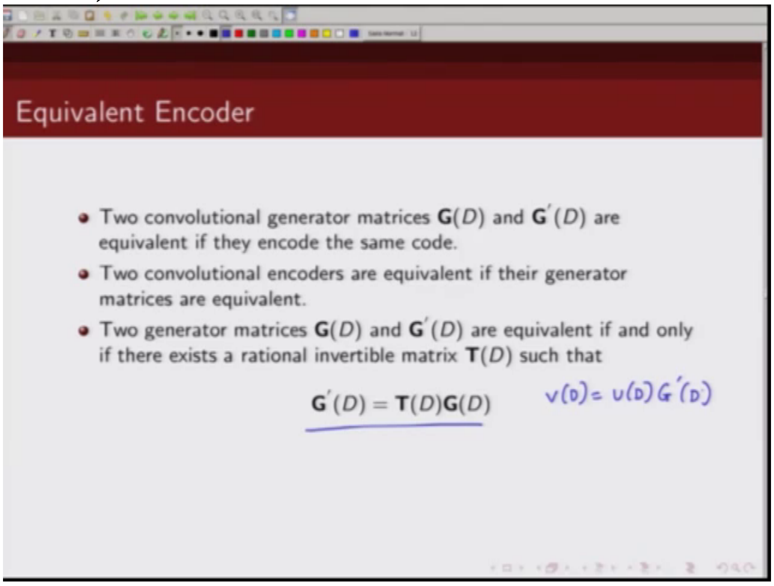
code then we say 2 convolutional encoders are equivalent. So if

(Refer Slide Time 15:18)



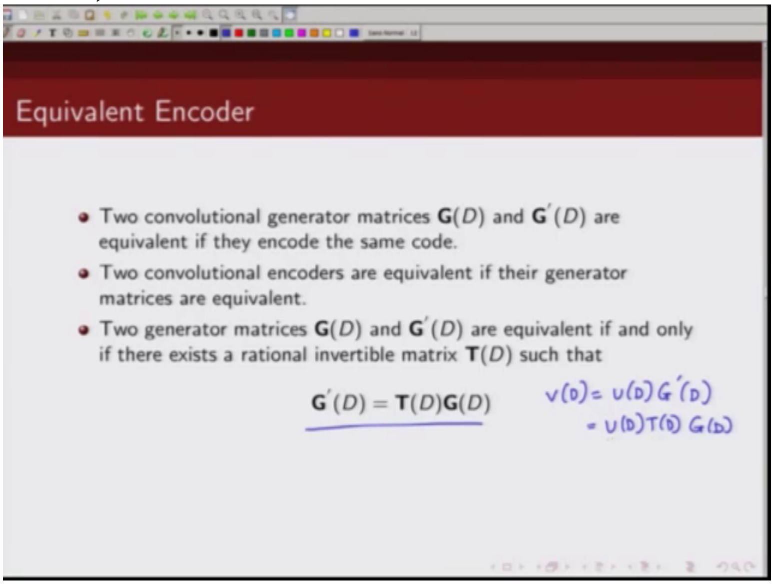
G D and G prime D are equivalent then this condition should hold. So two generator matrixes are equivalent if and only if there exists a rational invertible matrix T of D such that we can obtain G dash G by T D multiplied by G D, Ok and we can see basically so let's say set of codewords generated by G dash D. So that would be v D, u D times G dash D. Now this we can write

(Refer Slide Time 16:08)



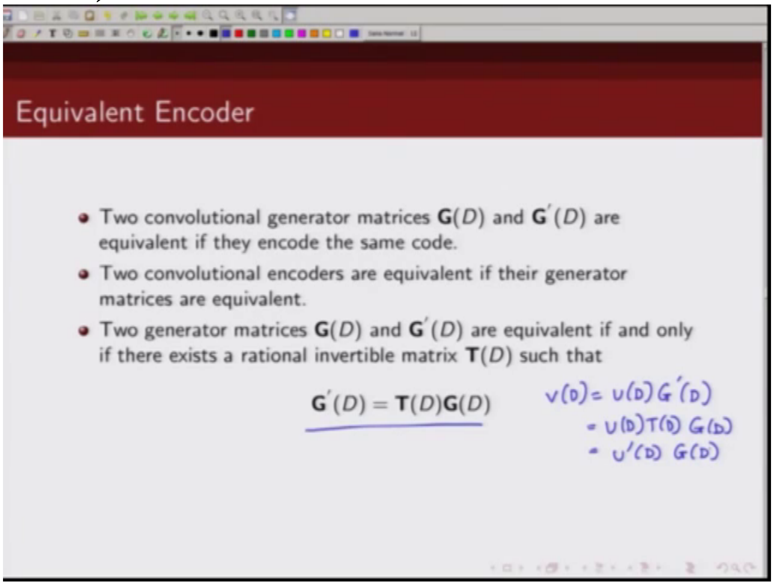
as u D T D times G of D and let us call

(Refer Slide Time 16:18)



u D T D as u dash D G D,

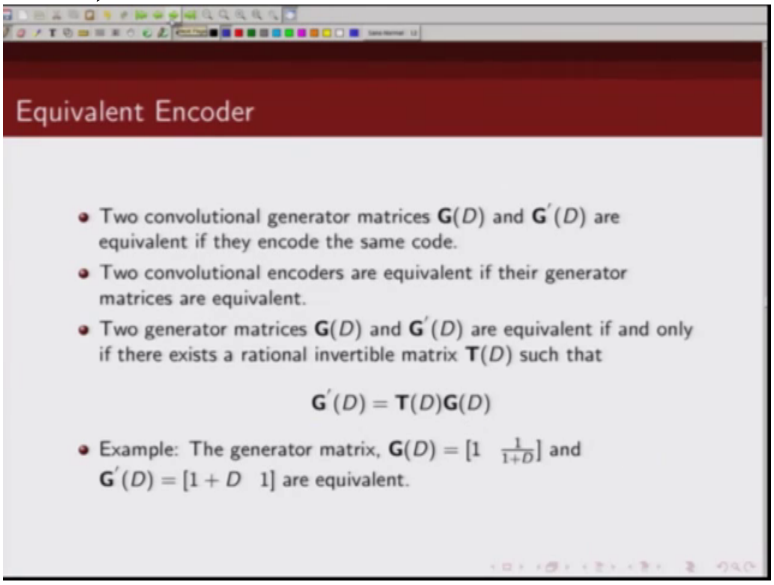
(Refer Slide Time 16:24)



Ok.

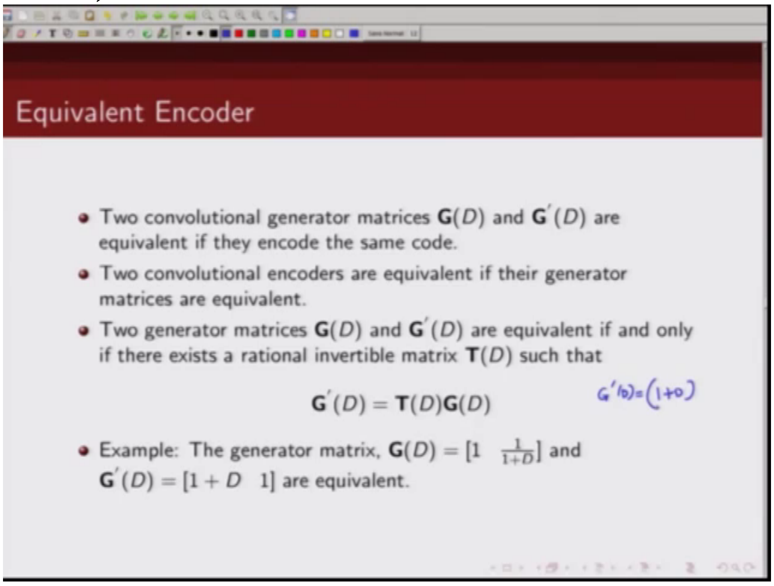
So let us take

(Refer Slide Time 16:29)



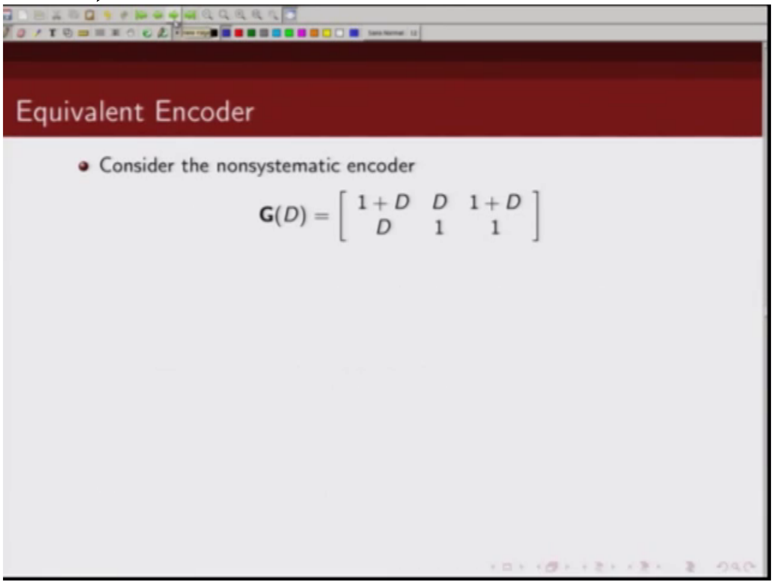
an example. This G d 1, 1 plus 1 by 1 plus D, and G dash D which is 1 plus D 1, these are equivalent encoders because we can write G dash of D as 1 plus D

(Refer Slide Time 16:52)



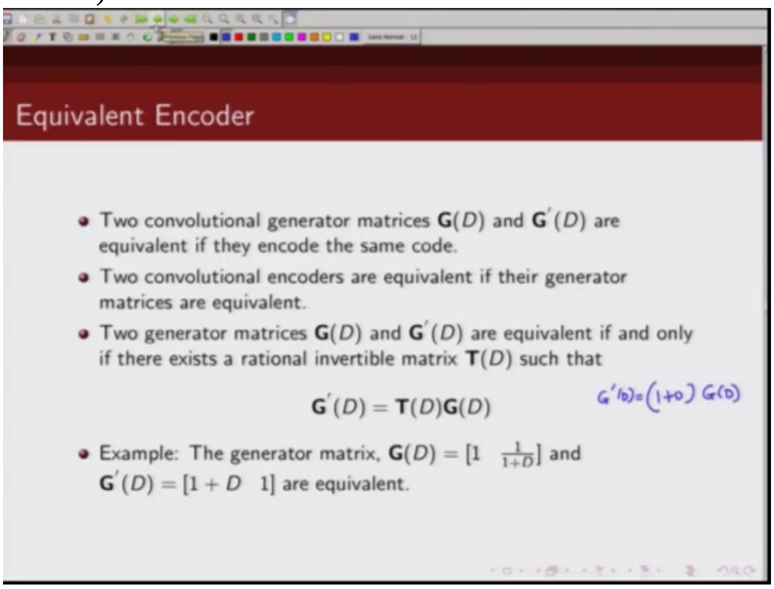
times G of D, Ok. And so for, and we can see this is a systematic encoder, generator matrix for a systematic encoder, Ok. Now for,

(Refer Slide Time 17:14)



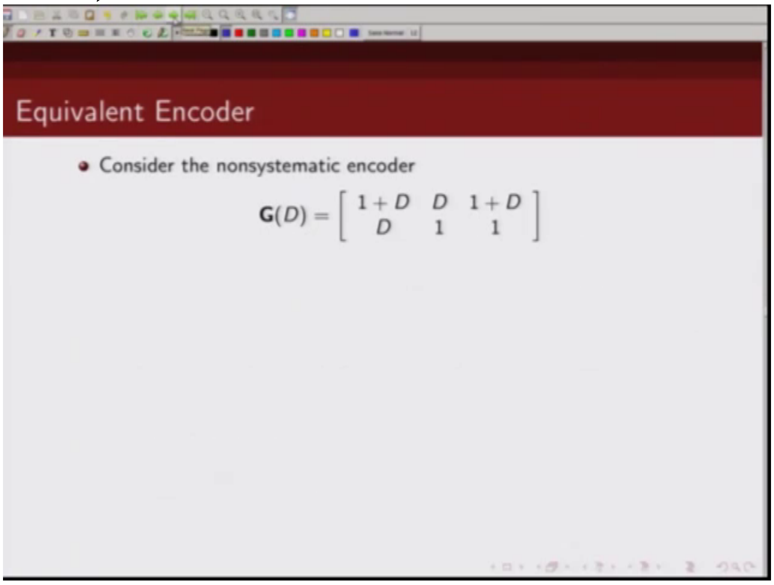
and this is

(Refer Slide Time 17:18)



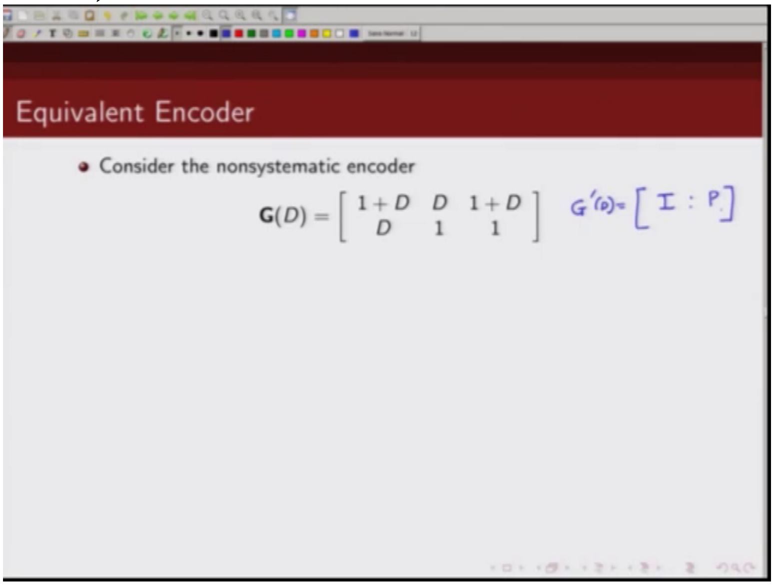
a feedback encoder, this is a feed forward encoder.

(Refer Slide Time 17:25)



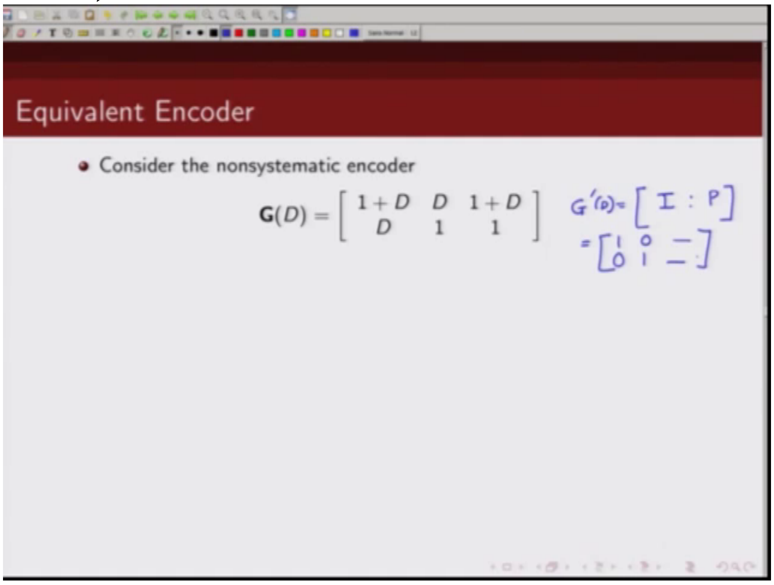
So let us take an example of a non-systematic feed forward encoder and let's try to find its equivalent systematic encoder. So what would be the equivalent systematic encoder corresponding to this non systematic encoder? The generator matrix G dash of D should be of the form identity and some matrix P.

(Refer Slide Time 17:56)



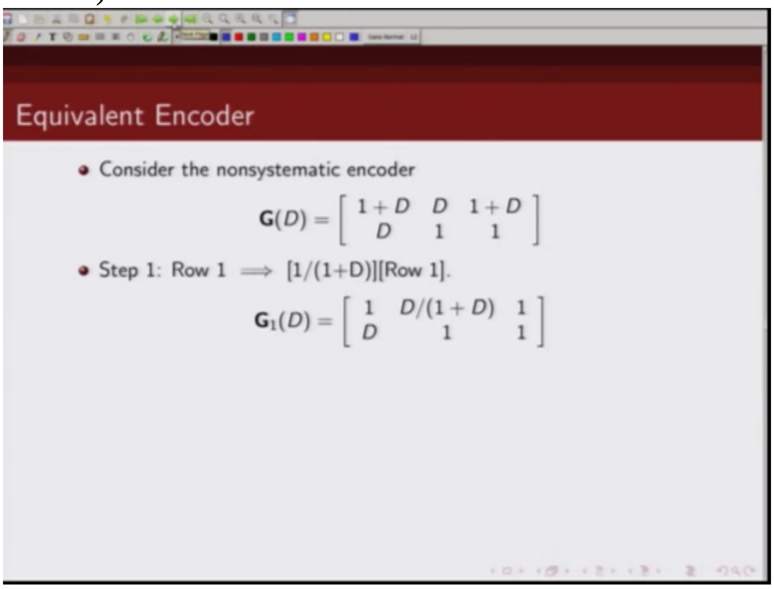
So what we want is basically we want this to be the form 1 0 something here 0 1 something here. So we want to convert this

(Refer Slide Time 18:10)



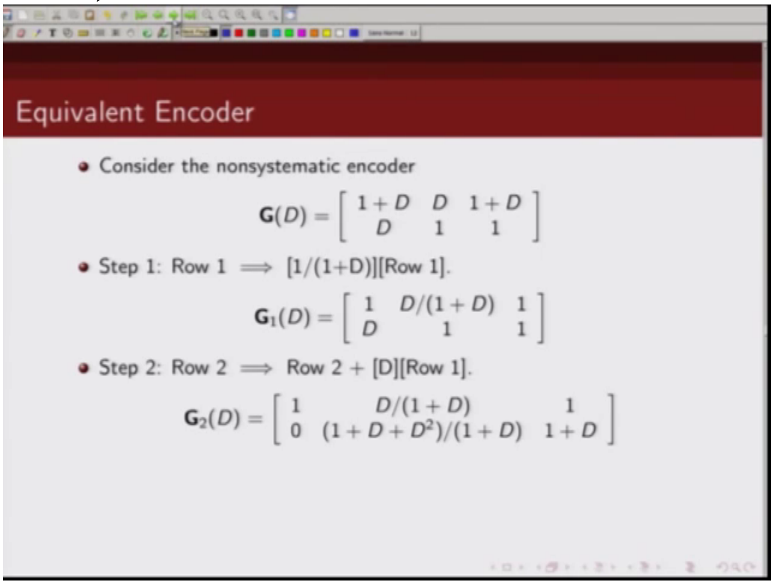
into a form of this type. Ok, so we will do elementary row operation to bring this generator matrix into a generator matrix of this form. So first,

(Refer Slide Time 18:28)



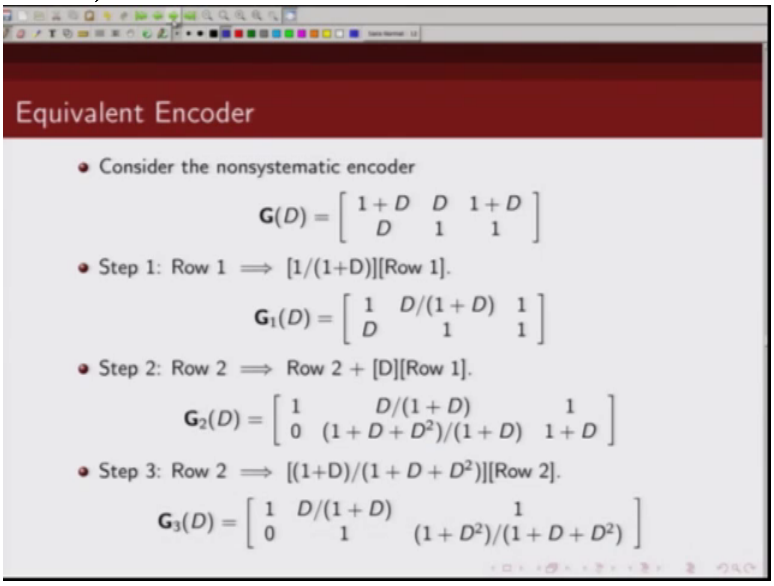
so we will do this transformation row 1, we will try to make this as 1. How can we make this as 1? If we multiply row by 1 by 1 plus D. If we do that, this term will become 1, this term will become D by 1 by D and this term will become 1. We leave the second row unchanged. Next, we want to get a

(Refer Slide Time 19:00)



zero here, right? How do we get a zero here? We do this transformation row 2; we will make it row 2 plus D times row 1. So the first row is unchanged but second row we do this transformation. It is row 2 plus D times row 1. So row 2 here is D plus D times row 1, which is another D. So D plus D is zero. Similarly row 2, this 1 plus D square plus 1 plus D, this is basically given by this and we have 1 plus D which is this term. So what we have done is we have converted this into form 1 0. Next we want to get a 1 here, right? We want to get the 1 here. How can we get a 1 here? We will do this following transformation.

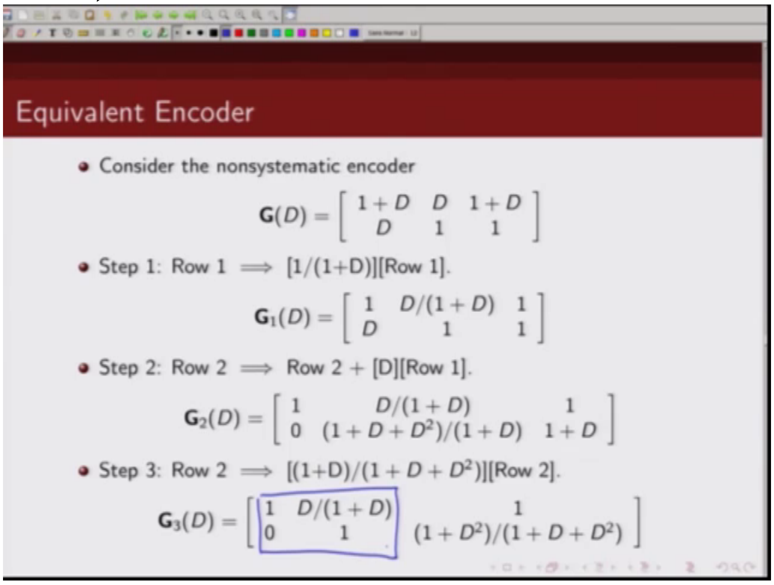
(Refer Slide Time 20:11)



For row 2, we will multiply row 2 by 1 plus D divided by 1 plus D plus D square. If we do that then this will become 1. So we leave the first row unchanged. Here 0, if we multiply by this, it does not change. If we multiply this by this we get a 1 here and here we get this term.

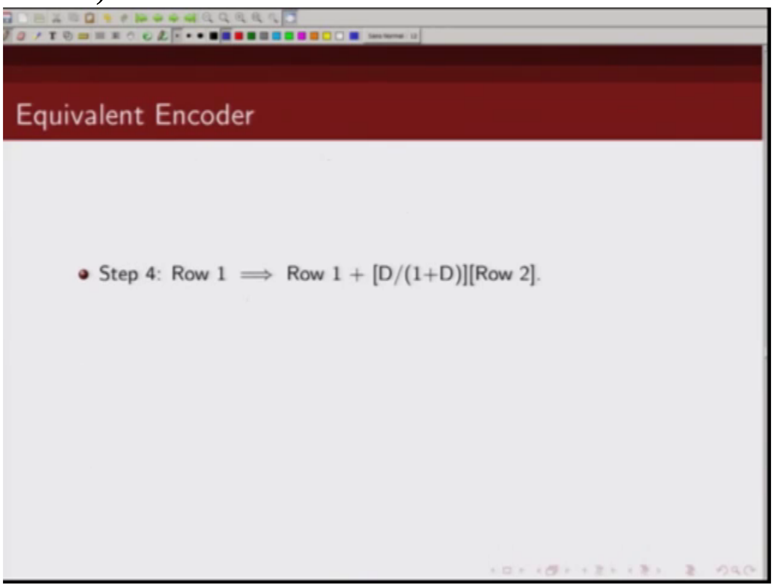
So now we have got, so far is we got a 1 here, we got a 0 here, we got a 1 here, now what else is remaining? We have to make this a, we have to make this a identity

(Refer Slide Time 20:53)



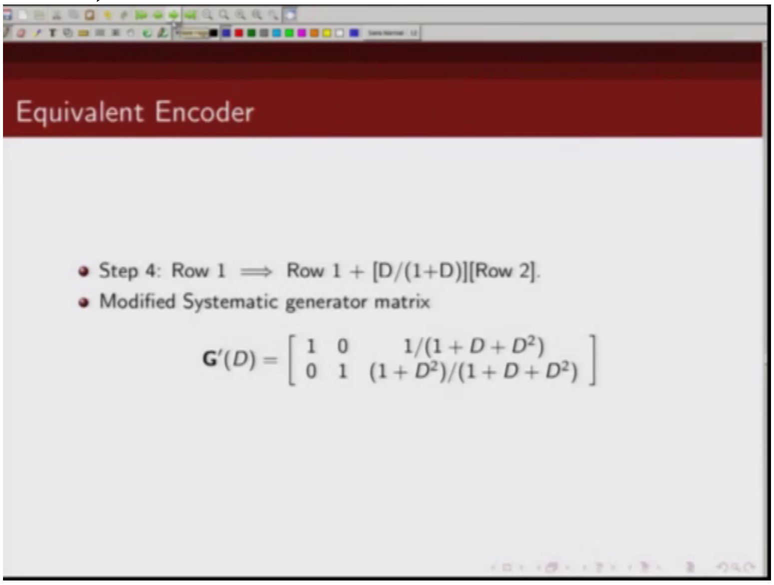
matrix. So we have to make this a zero. How can we make this a zero? We multiply this by this and add it up to the first row. We can make it a zero. So next,

(Refer Slide Time 21:09)



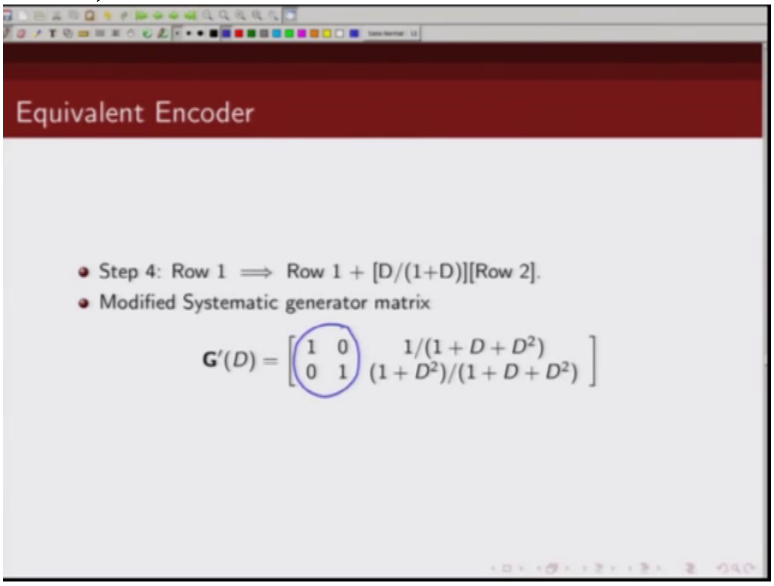
row 1 we add D times 1, 1 plus D times row 2. If we do that

(Refer Slide Time 21:18)



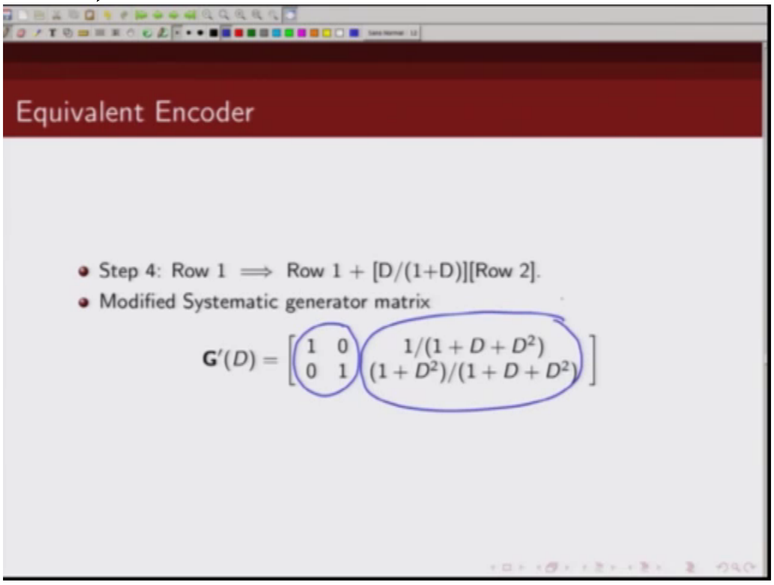
the modified generator matrix that we get is this. Note now this is generator matrix for a systematic encoder. You have your identity matrix here. And you have some matrix

(Refer Slide Time 21:31)



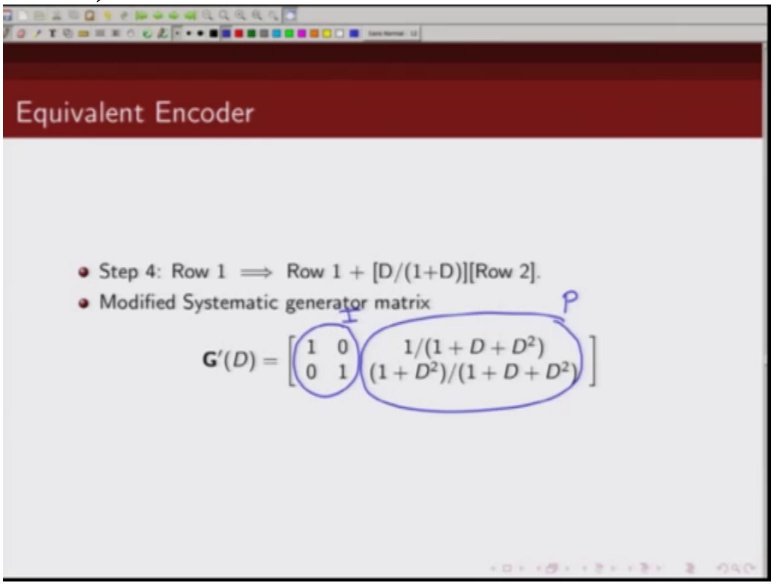
here which is your P matrix.

(Refer Slide Time 21:34)



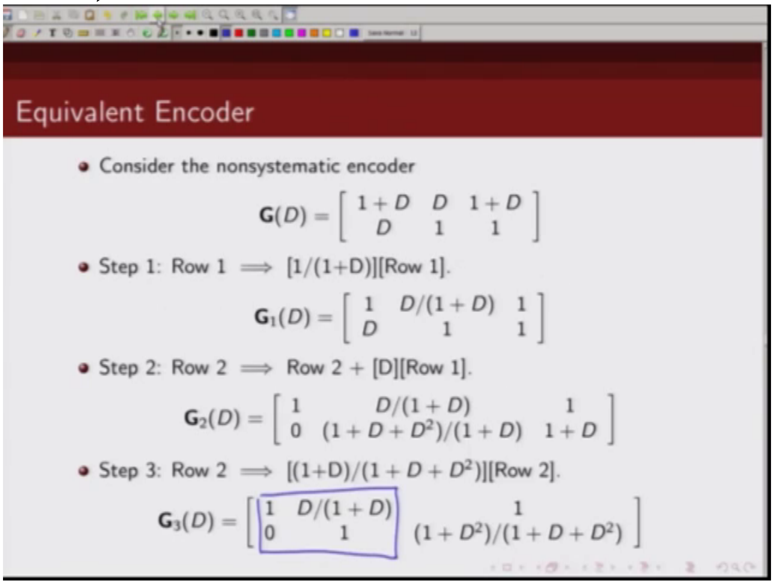
So this is basically

(Refer Slide Time 21:37)



the generator matrix for systematic encoder. So note now by

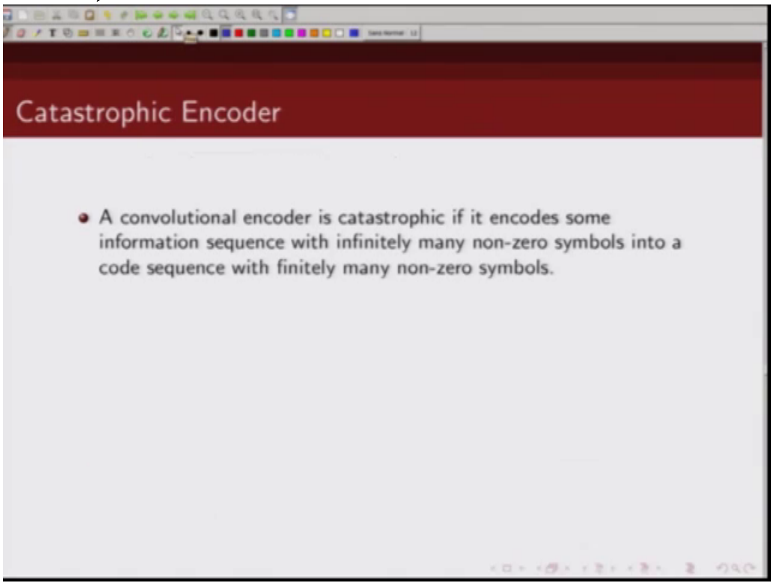
(Refer Slide Time 21:46)



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

Next

(Refer Slide Time 22:02)

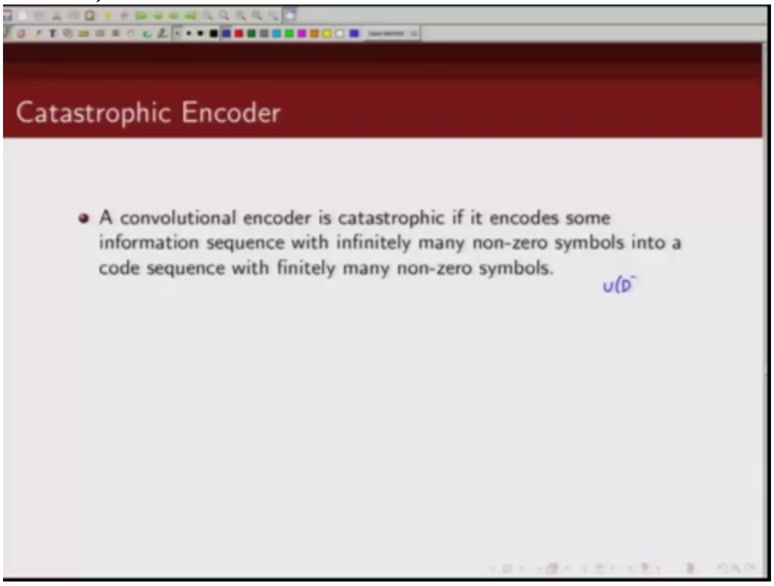


we will explain the concept of catastrophic encoder. So convolutional encoder is catastrophic if it encodes some information sequence which has large weight, which has large number of 1's into a code sequence with finite number of 1's. So if you have an information sequence of let's say,

(Refer Slide Time 22:26)

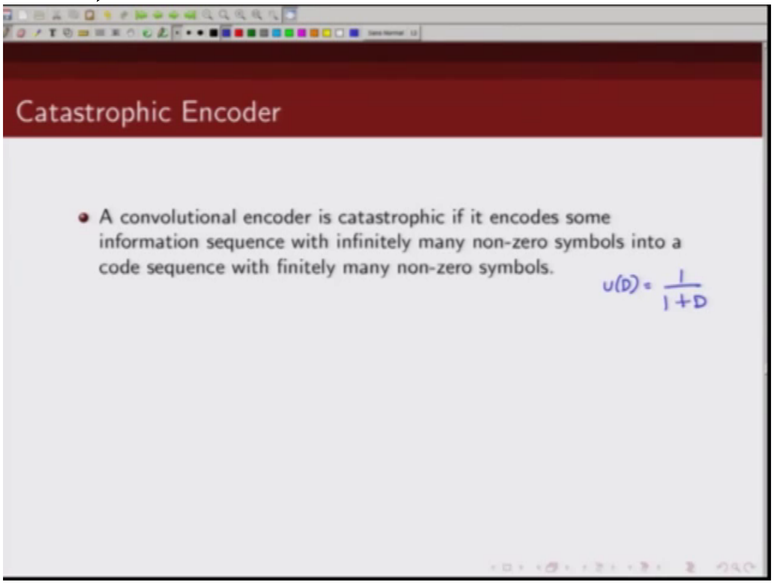
u of D which is

(Refer Slide Time 22:29)



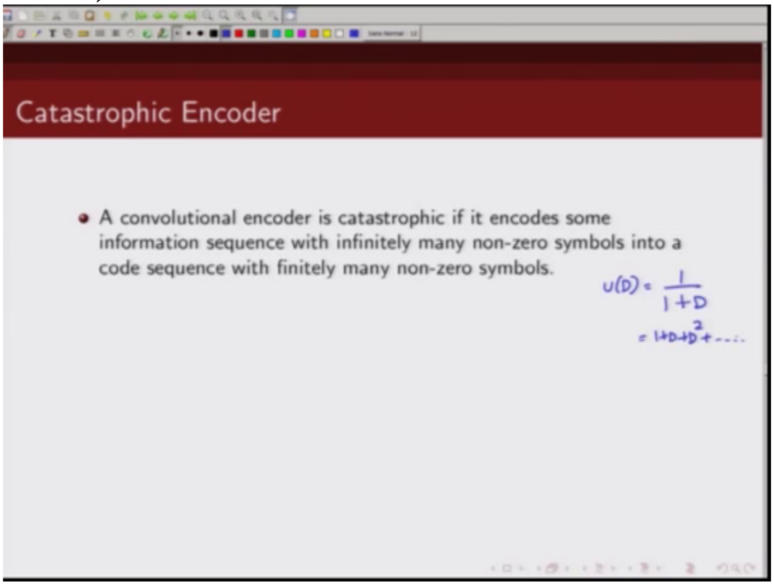
1, 1 plus D. Now this is a sequence of all 1's.

(Refer Slide Time 22:33)



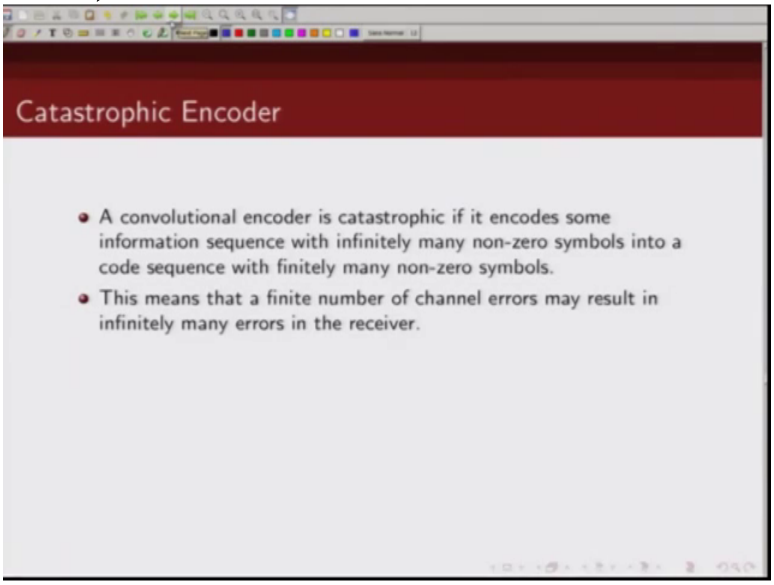
This is basically nothing but 1 plus D plus D square dah dah dah, so this is a sequence of

(Refer Slide Time 22:41)



1, all 1's. Now if you have an encoder which maps a sequence, input sequence which has large number of 1's into a sequence, coded sequence with finite number of 1's; now that type of encoder is known as catastrophic encoder.

(Refer Slide Time 23:04)

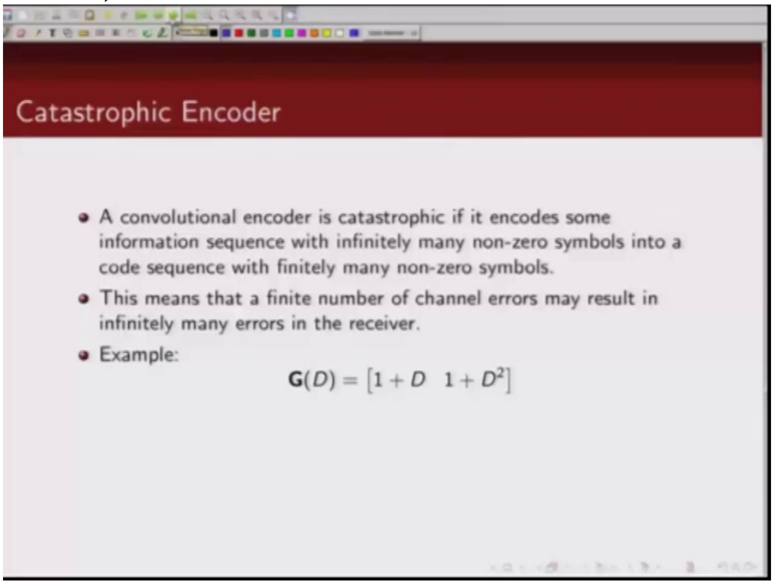


Now why is it catastrophic? So to illustrate it, we will take an example. It is an catastrophic encoder because a finite number of channel errors can result in infinite number of input errors. Because you have your information sequence which has large number of 1's,

(Refer Slide Time 23:25)

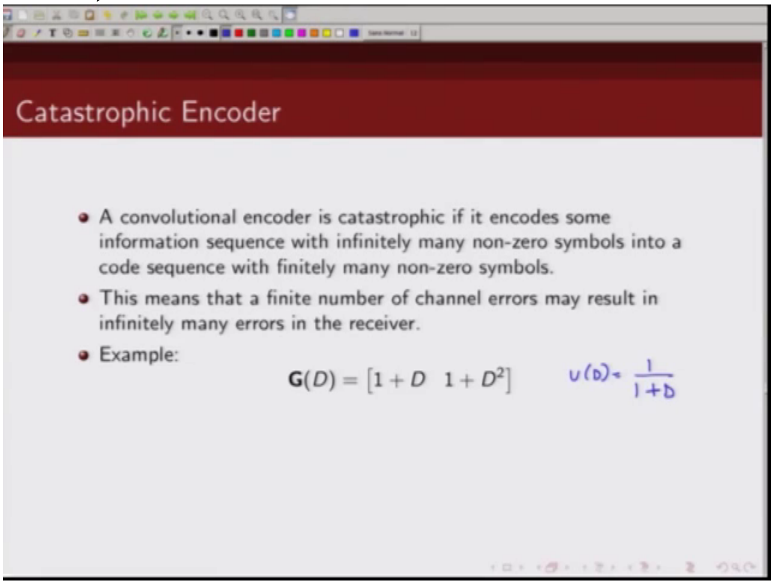
possibly infinite number of 1's. Because that information sequence is getting mapped, coded sequence with finite number of 1s; if error happens in those locations where you have finite number of 1's then your output sequence will get transformed into an all zero sequence and your decoder will think you have transmitted an all zero sequence; whereas actually you had transmitted a sequence of all 1s'. So finite number of channel errors in case of a catastrophic encoders can result in infinite number of input errors.

(Refer Slide Time 24:05)



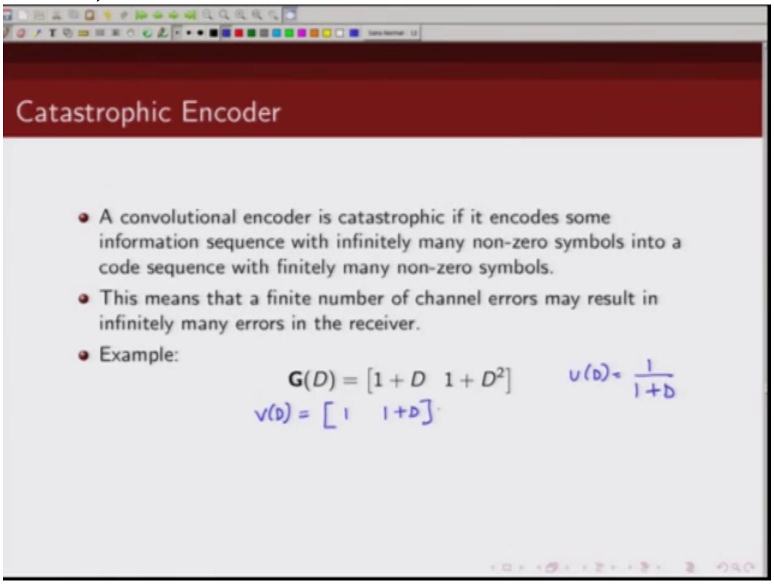
Let us take an example of this encoder with generator matrix G D which is given by 1 plus D and 1 plus D square and let us feed input which is all, sequence of all 1's which I can write as 1

(Refer Slide Time 24:23)



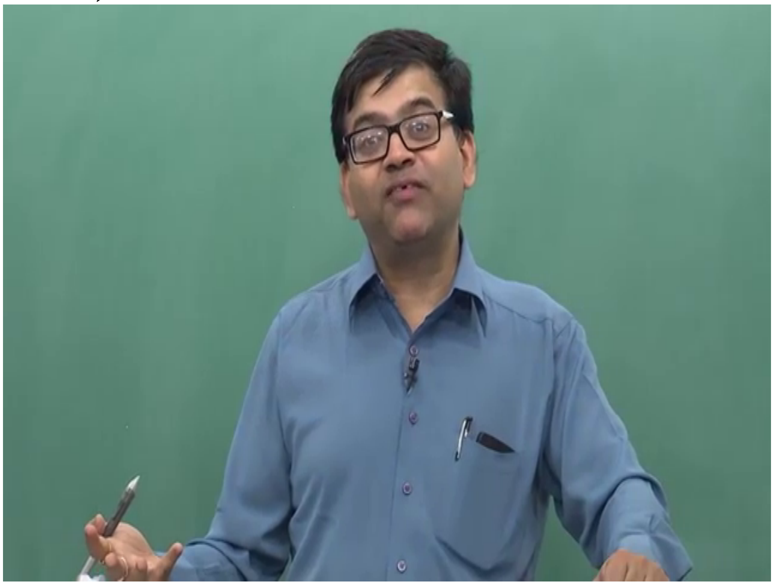
by 1 plus D. Now if this information sequence passes through this encoder what would be your output sequence? Output sequence would be 1 and this would be 1 plus D. So what you would get is

(Refer Slide Time 24:41)



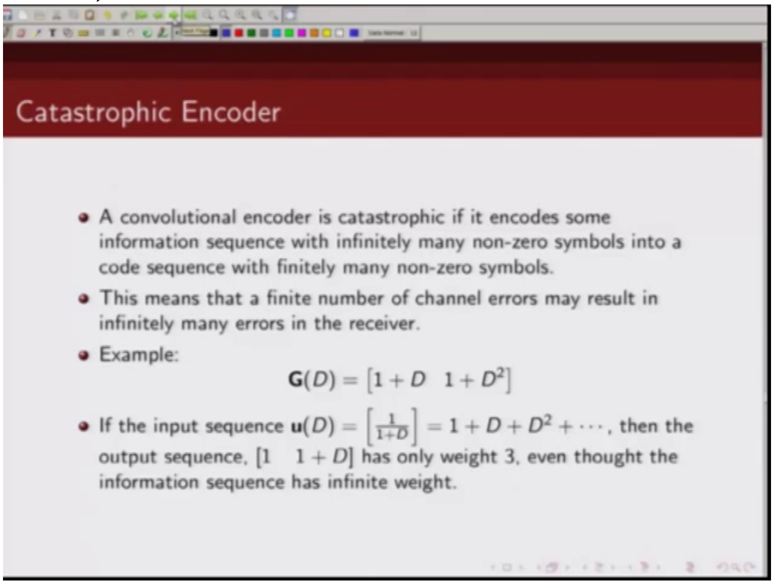
you would get output sequence which has weight only 3 where as information sequence has infinite number of 1's. So here is an example where an input sequence of very large number of 1's getting mapped to an output sequence of only weight 3. What if error happens in these 3 locations where you had 1's? Then your output sequence that

(Refer Slide Time 25:13)



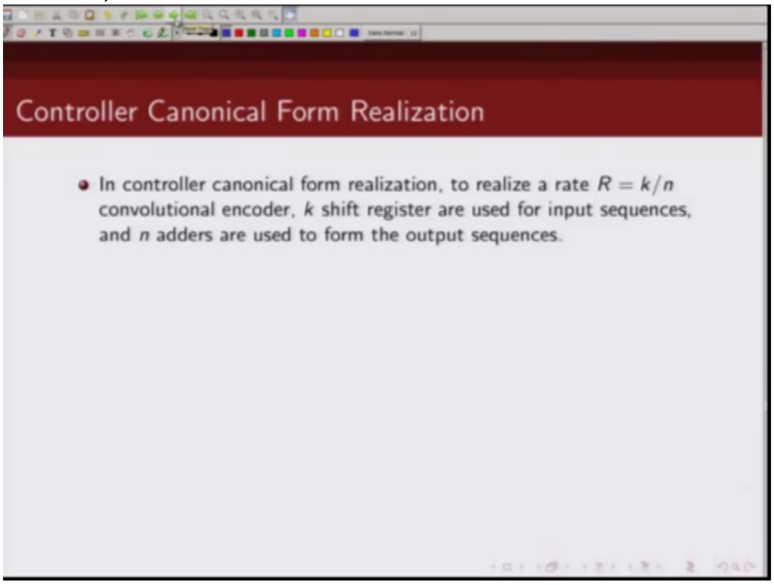
decoder will, receiver will receive will be all zero sequence and the receiver will think that you transmit, you transmitted all zero sequence where as the input is all 1 sequence. So you can see in case of a catastrophic encoder, a finite number of errors, in this example only 3 errors can result in infinite number of errors, input errors Ok.

(Refer Slide Time 25:43)



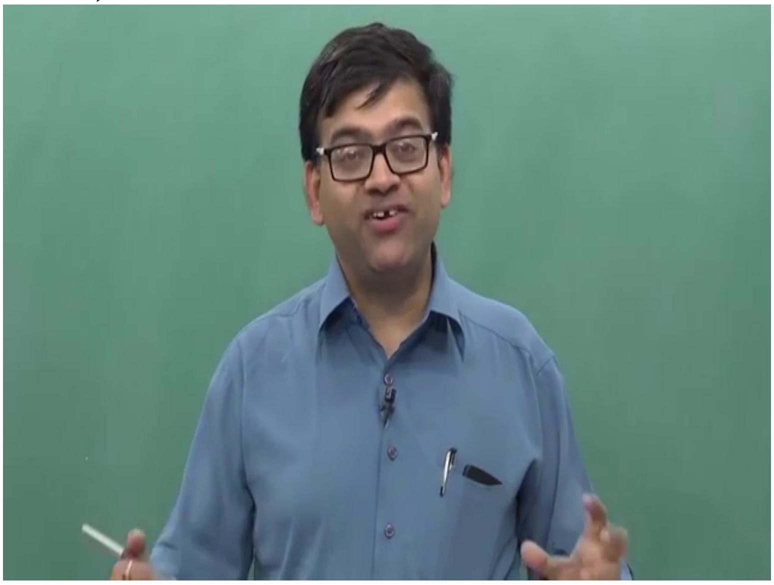
This I have explained.

(Refer Slide Time 25:47)



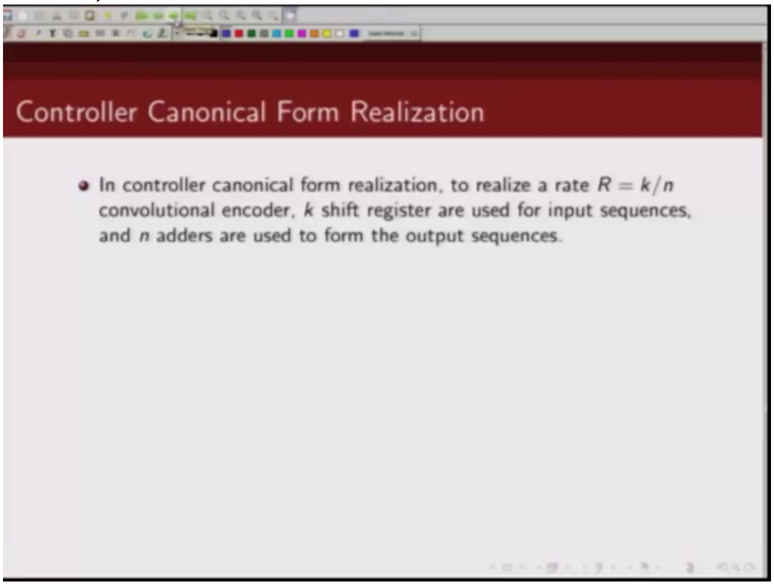
Next I am going to come to the topic of realization of a convolutional encoder. How can we represent

(Refer Slide Time 25:57)



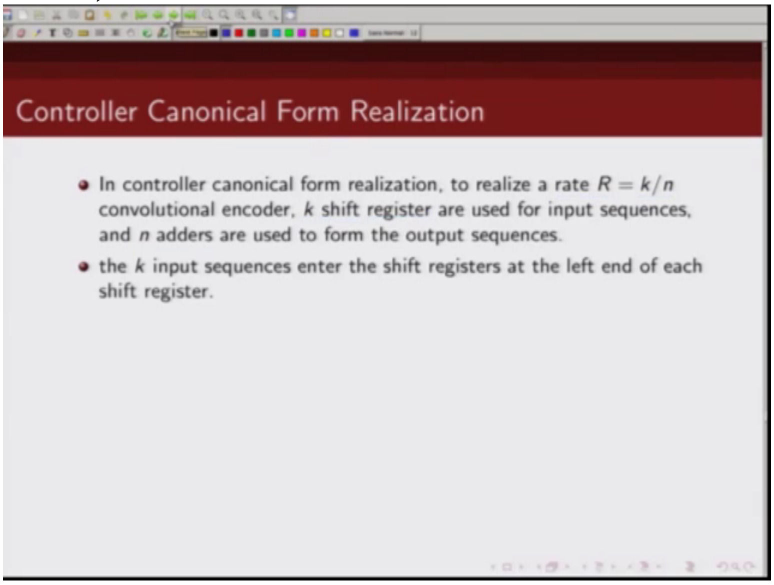
a convolutional encoder using shift register? So given a generator matrix how can you implement a convolutional encoder? So in this we are going to talk about 2 such type of realization. The first one that we are going to discuss now is known as controller canonical form realization. So in a controller canonical

(Refer Slide Time 26:20)



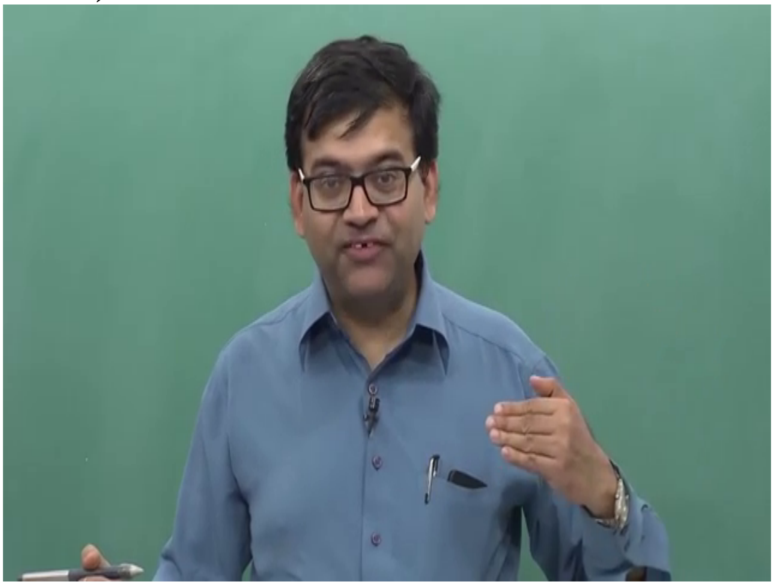
form realization if you have a rate k by n convolutional encoder we use k shift registers. So the number of shift registers used is equal to number of information sequence that you have. And the output is obtained by using n set of adders, one for each output sequence.

(Refer Slide Time 26:54)



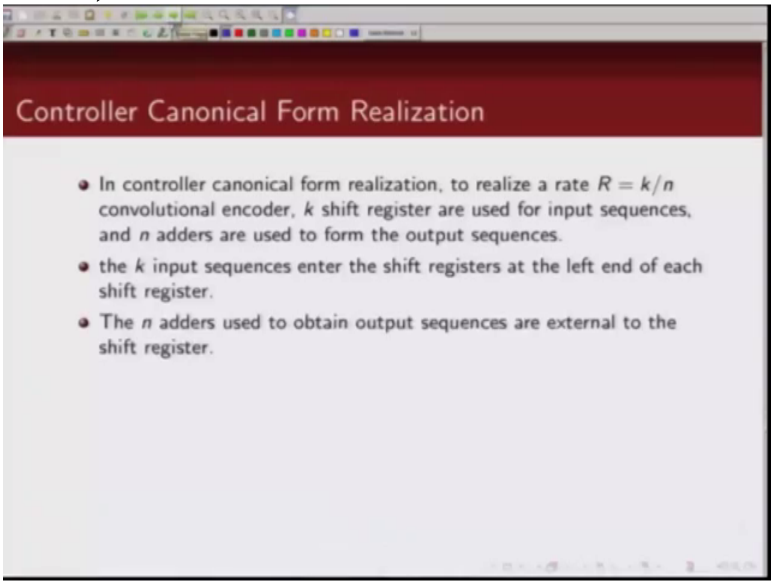
And in this case the key input sequence enter the

(Refer Slide Time 26:59)



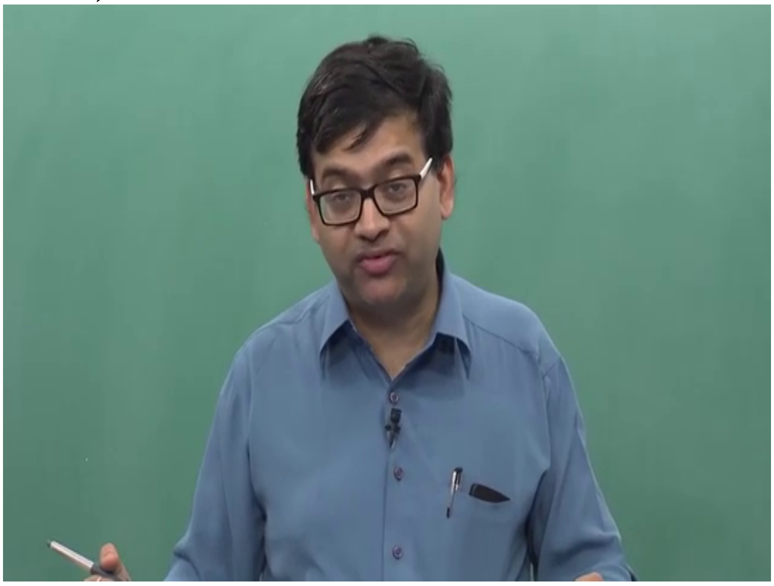
shift register from the left hand side and we take the output from the right hand side.

(Refer Slide Time 27:05)



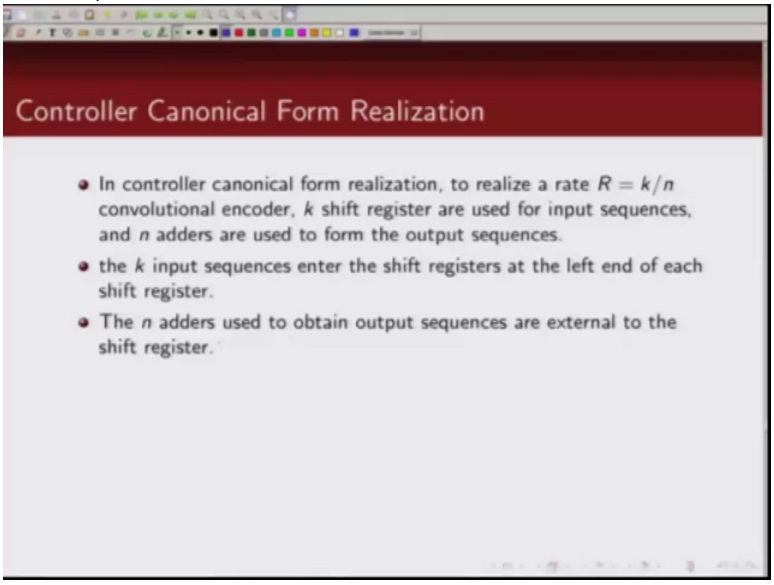
The next point to remember here is in case of a controller canonical form realization; these n adders that are used to obtain the output sequence, the

(Refer Slide Time 27:15)



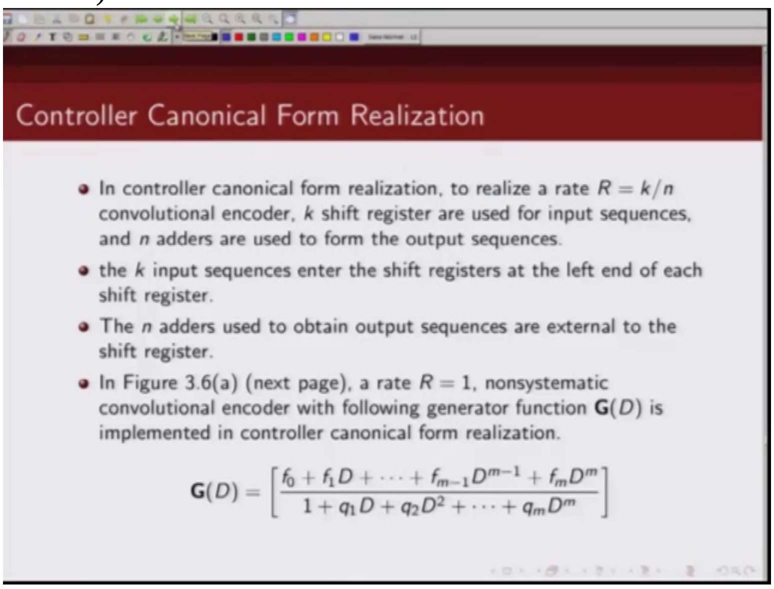
coded sequence, these adders are external to the shift registers. So they are not inside

(Refer Slide Time 27:22)



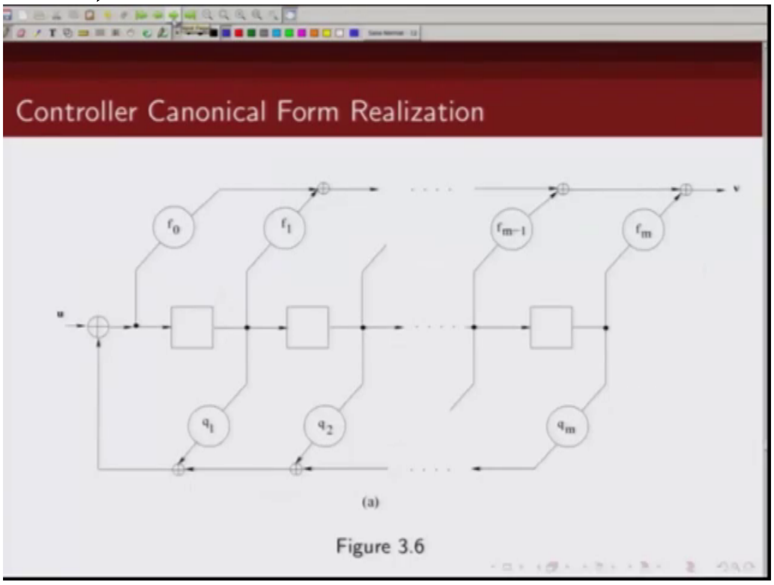
the shift registers.

(Refer Slide Time 27:25)



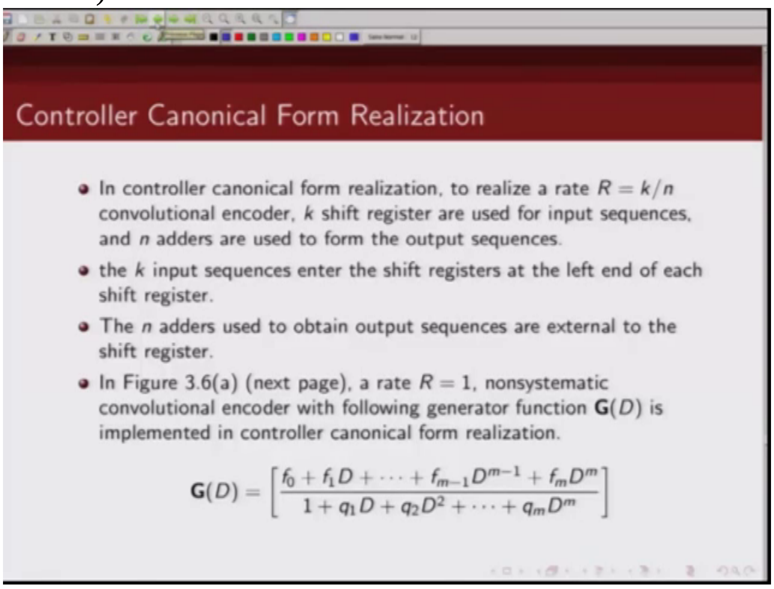
So let us take an example of a rate 1 non systematic convolutional encoder whose generator matrix is given by this. So in the numerator you have f 0 plus f 1 D plus f 2 D square like that. Similarly denominator you have 1 plus q 1 D plus q 2 D square like that. So how can we implement this using controller canonical form realization? So let's go back. So we are going to use k shift registers. So this is rate 1, 1 by 1 so there will be only 1 shift register. So we use

(Refer Slide Time 28:07)



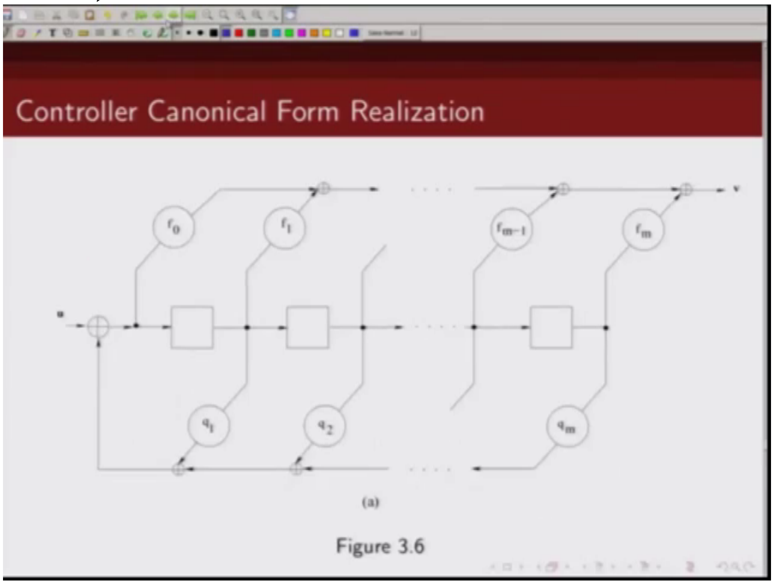
1 set of shift register corresponding to 1 input sequence. Next

(Refer Slide Time 28:16)



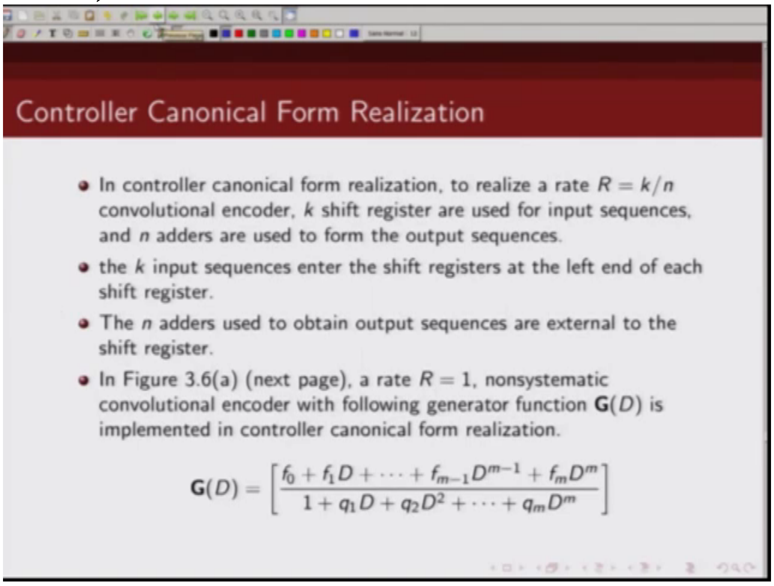
we use n set of adders. Now what is n here? Because it is rate 1, so n is also 1. So we will use 1 set of adders.

(Refer Slide Time 28:24)



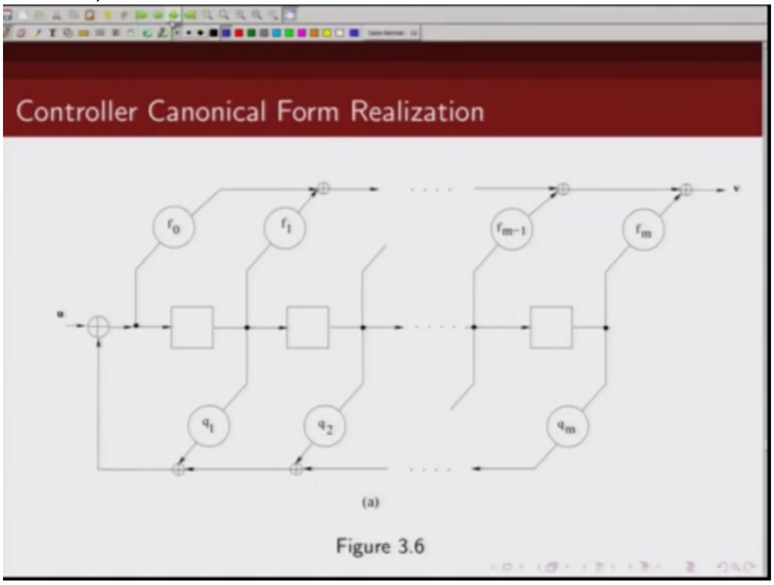
And these set of adders basically, this output that we are seeing, we have this n set of uh, adders that we are using to obtain this coded sequence v. Now

(Refer Slide Time 28:38)



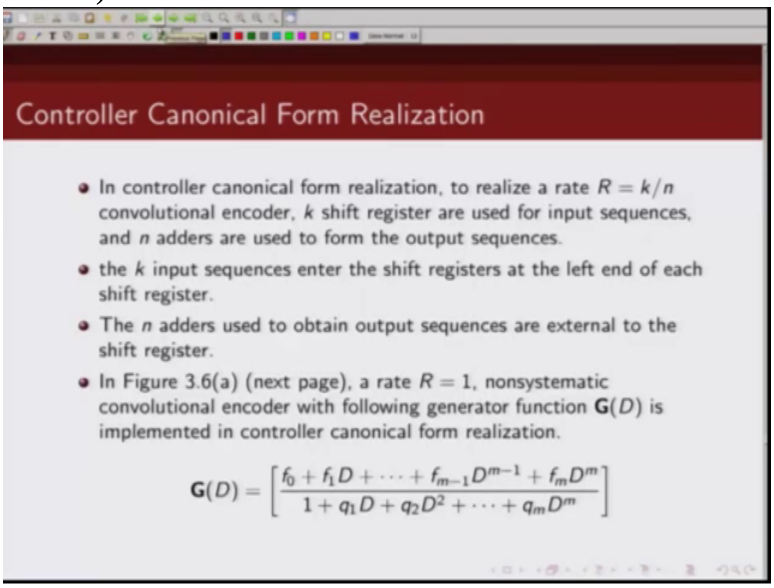
the key input sequence enter the shift register from the left hand side; so we can see here

(Refer Slide Time 28:46)



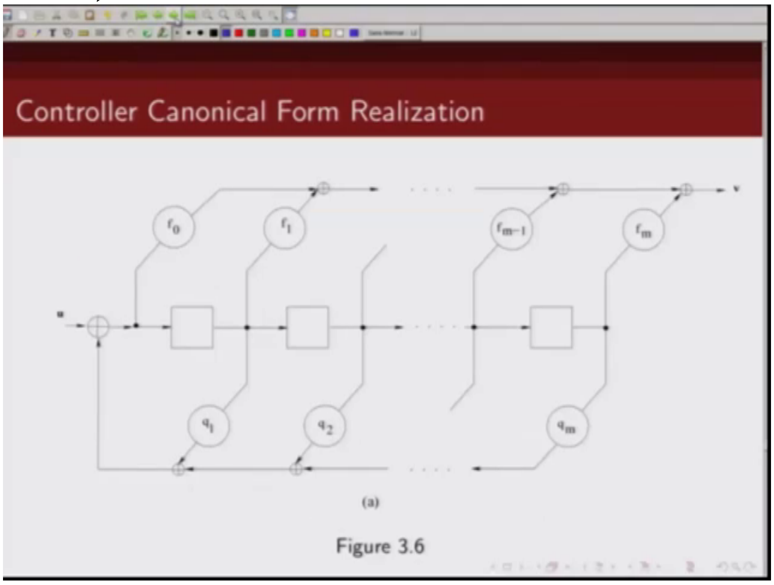
the input is entering from this (()) side. So since this is a feedback encoder so let's

(Refer Slide Time 28:52)



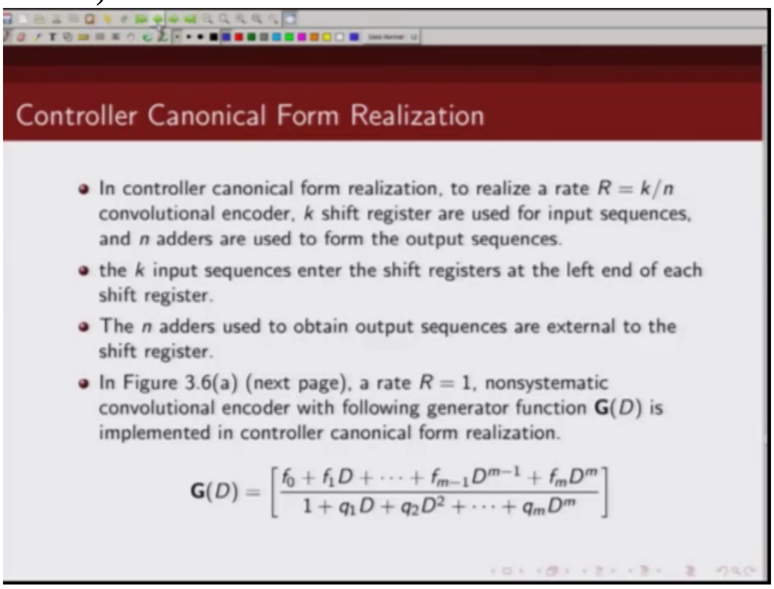
first look at the numerator term. What do we have here? We have f 0 plus f 1 D plus f square D so this input is basically multiplied by f 0. So current input is

(Refer Slide Time 29:08)



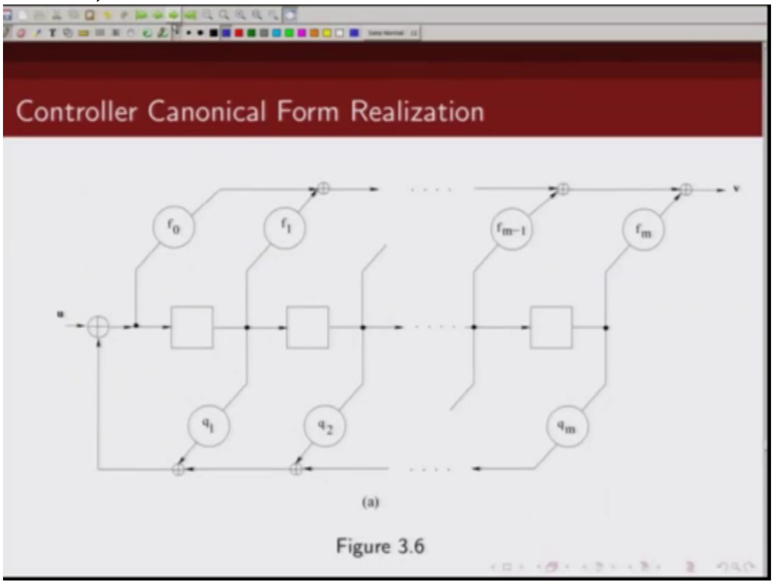
getting multiplied by f 0. Then one delayed version of input is getting multiplied by f 1, 2 delayed version is getting multiplied by f 2. So you can see this is then f 0, this is f 1, this is f 2 and again whether there is a connection from this input to the output, depending on that either f 0, f 1, f 2 will be either 1 or 0. If there is a connection, this will be 1, if there is no connection, this will be zero. So you can see this is f 0, this is f 1 D, f 2 D square like that basically if this is mth delay element this will be f m D m. Similarly you look, go back and look

(Refer Slide Time 29:59)



at the denominator. We have 1 plus q 1 D plus q 2 D square like that, so

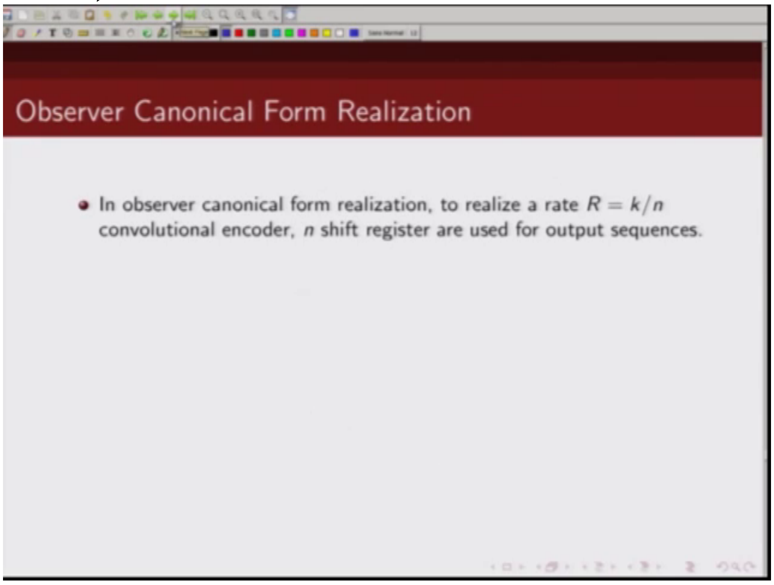
(Refer Slide Time 30:07)



this is the input 1, this is, this is the D term q 1 D term, so this multiplied by q 1, this is D square term multiplied by q 2, like that and then finally you have D m term which is getting multiplied by q m. So you can see this is how we can realize convolutional code using controller canonical form realization. Please note these adders are external to shift register. There are no adders here internal to shift registers. The inputs are entering on left hand side where as output is taken from right hand side.

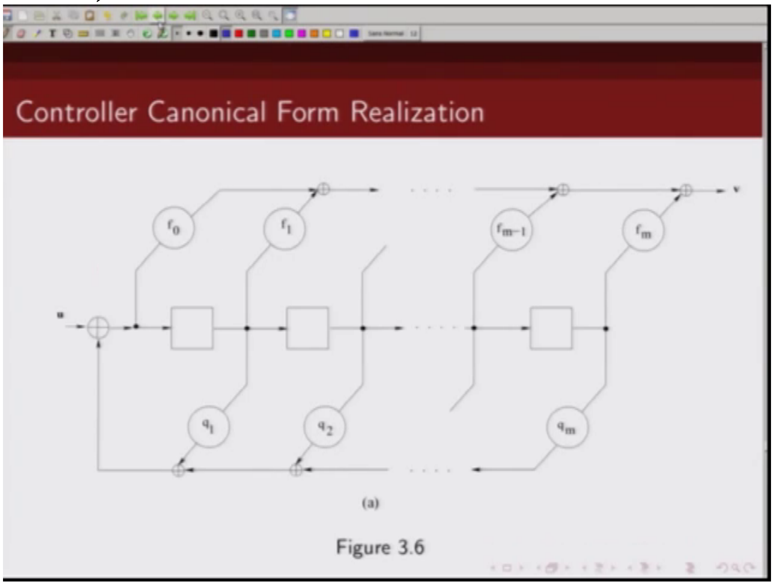
Now contrast this with

(Refer Slide Time 30:54)



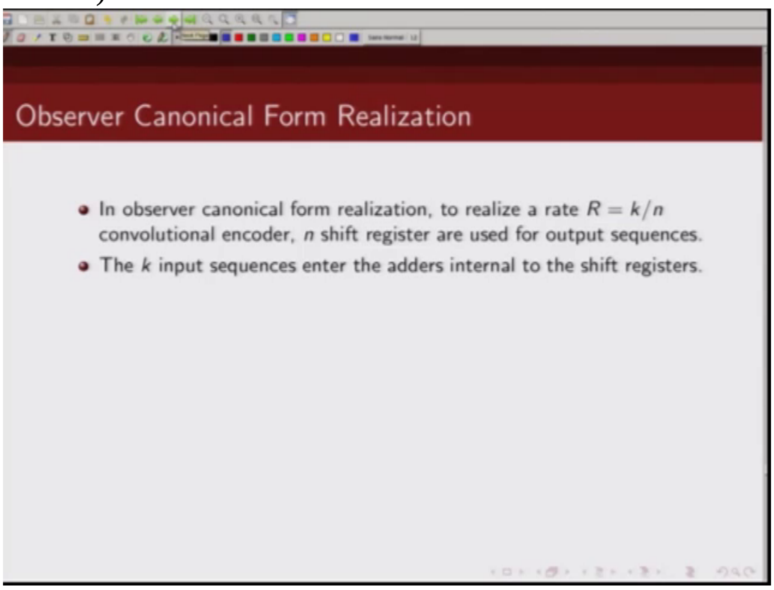
observer canonical form realization. So now observer canonical form realization, we need to realize the rate k by n encoder we require n shift registers. Now please note for the controller canonical form realization we require k set of shift registers; where as in this case we require n set of shift registers, one for each of the coded bits. The second difference is k input sequences in the observer canonical form realization, these k sequences enter into the shift register and these adders are internal to the shift registers. If you recall in case of a controller canonical

(Refer Slide Time 31:49)



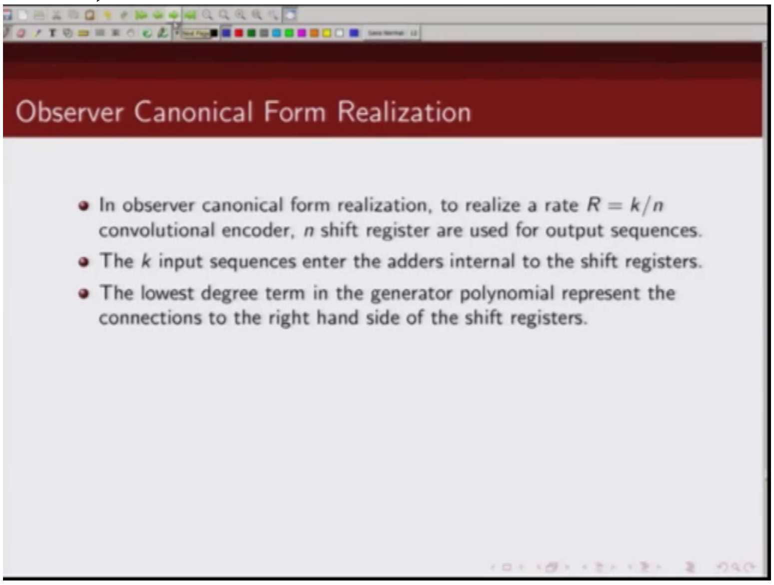
form realization, the input is entering here. And at each time instance when your clock comes they move, they shift to one location to the right. This will move to here, this will move to here where as in the observer canonical

(Refer Slide Time 32:04)



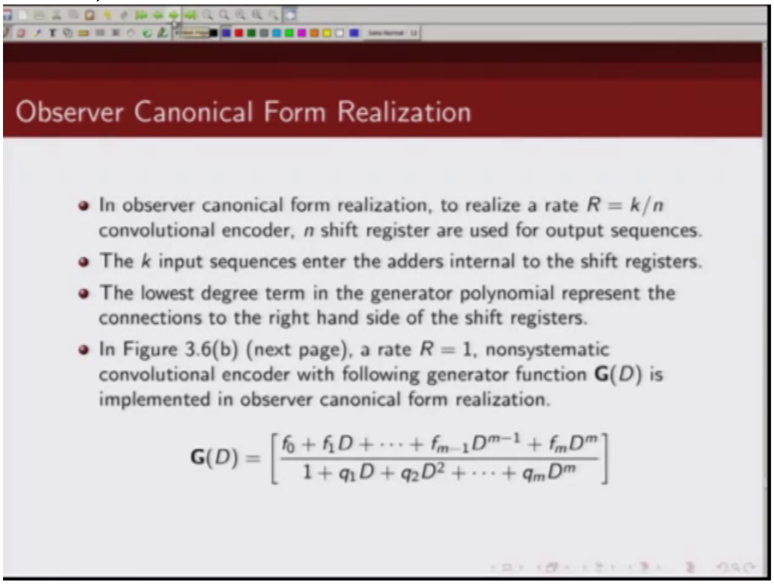
form realization these inputs are directly entering into the shift register and these adders are internal to the shift register. We will give an example to illustrate what we mean.

(Refer Slide Time 32:18)



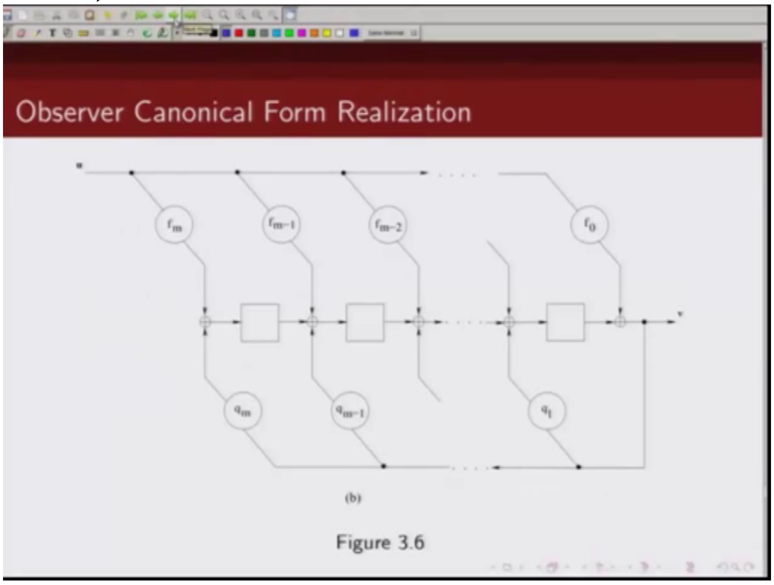
The lowest degree term generator matrix represents the connection to the right hand side of the shift register. In case of controller canonical form realization the lowest degree term was on the left hand side. Here the lowest degree term will be on the right hand side. So

(Refer Slide Time 32:39)



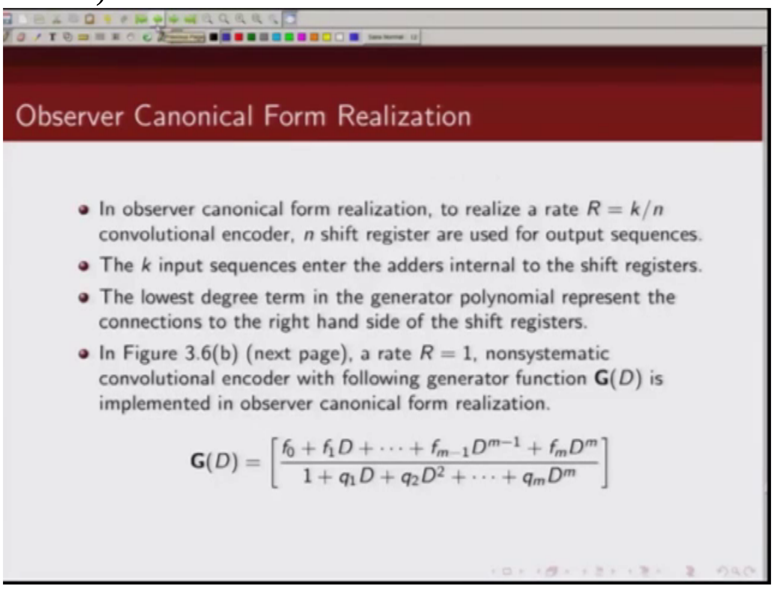
let us take the same example that we considered earlier. So we are considering the same generator matrix and we are going to realize generator matrix now using observer canonical form realization. So again here k is 1, n is 1, so we have n is1, so we have one set of shift registers.

(Refer Slide Time 32:59)



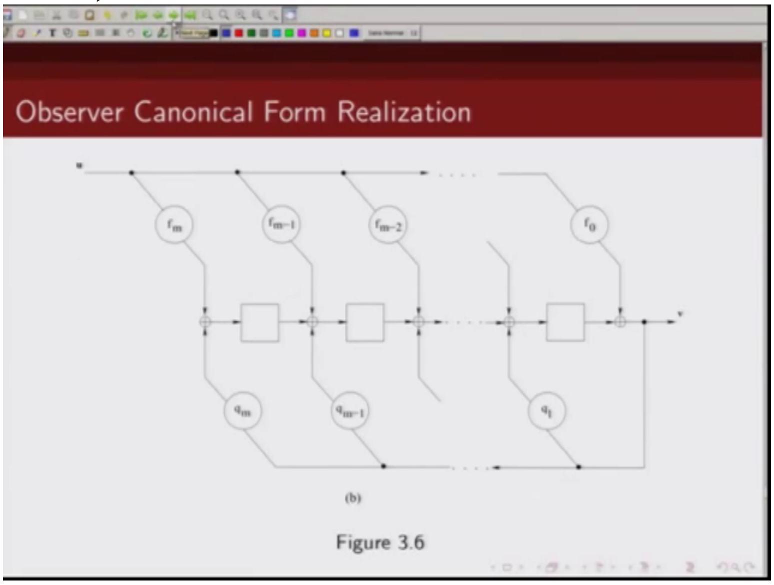
This is one set of shift registers. Next what did we say,

(Refer Slide Time 33:04)



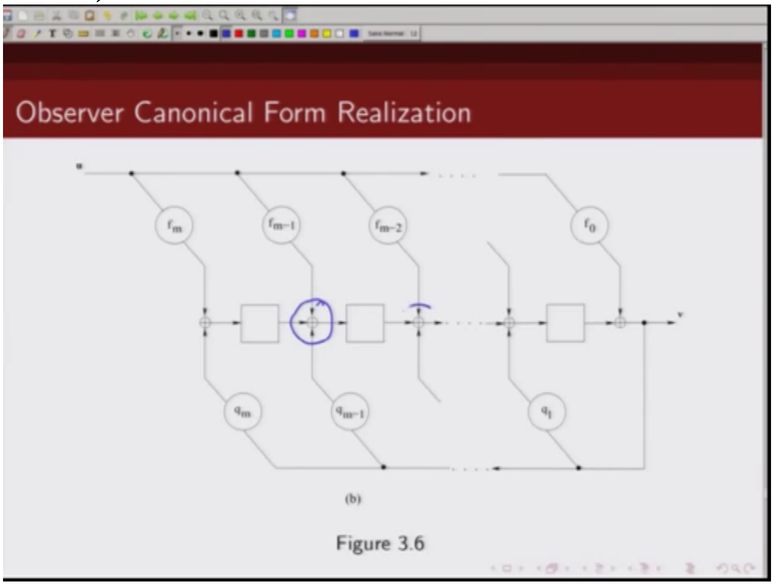
the k input sequence enter the adder internal to shift register and what do we mean by internal? So these are shift register elements,

(Refer Slide Time 33:13)



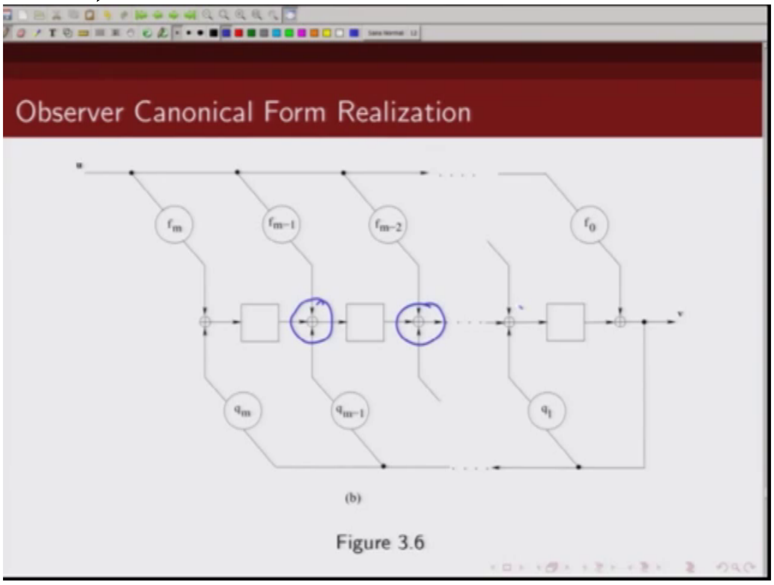
delay elements and note these adders are in between

(Refer Slide Time 33:19)



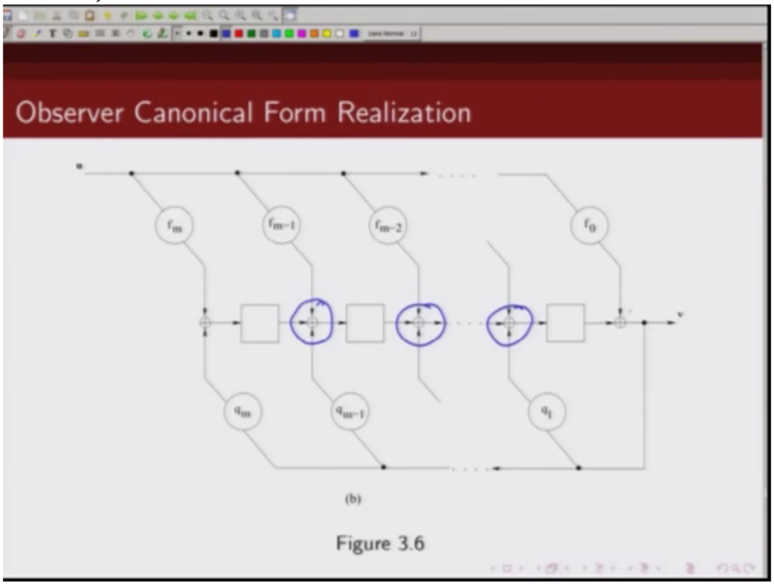
the

(Refer Slide Time 33:20)



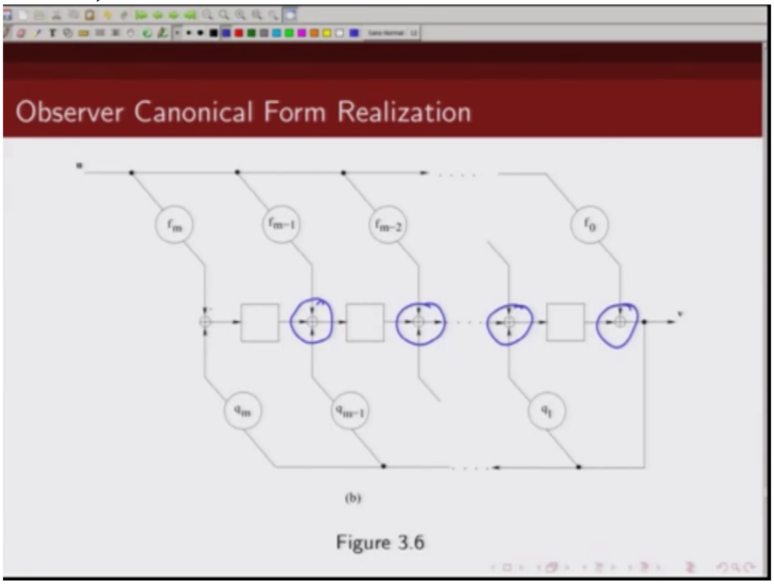
shift registers. These

(Refer Slide Time 33:21)



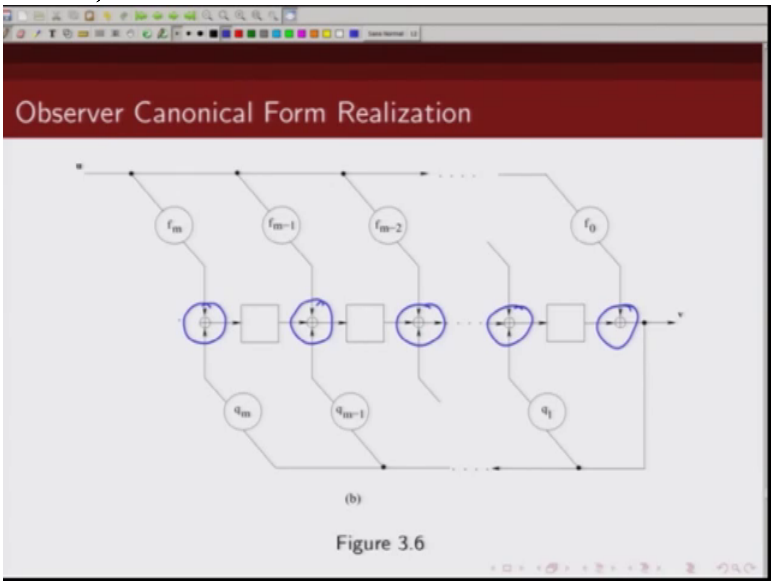
adders are internal

(Refer Slide Time 33:22)



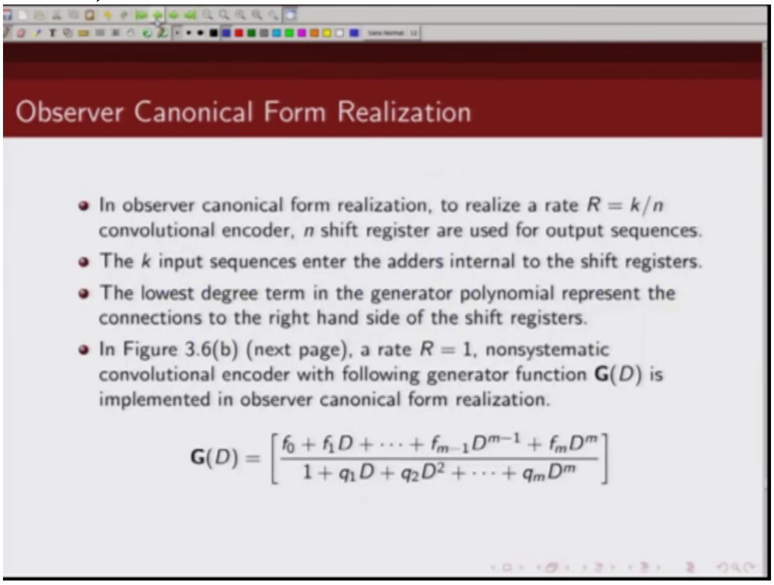
to the shift registers, Ok

(Refer Slide Time 33:27)



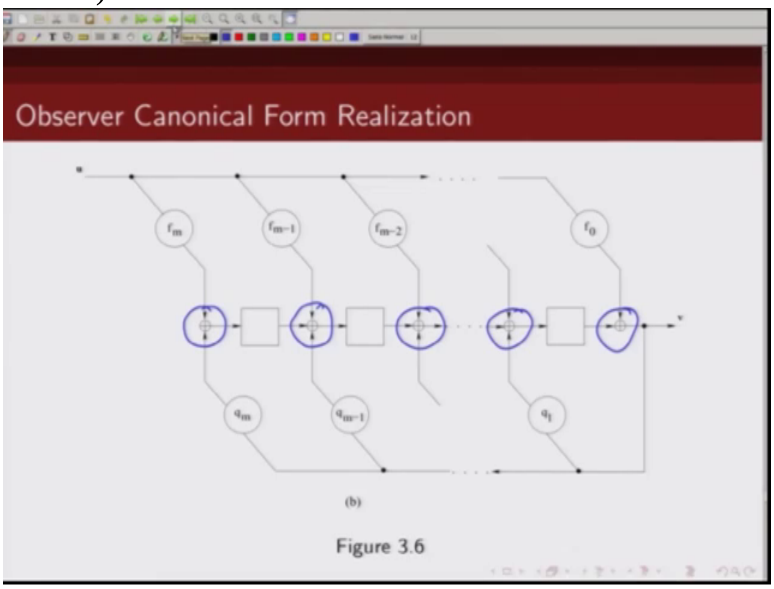
and next thing that we said

(Refer Slide Time 33:29)



was the lowest degree term in the generator matrix represents connection to the right hand side. You see here,

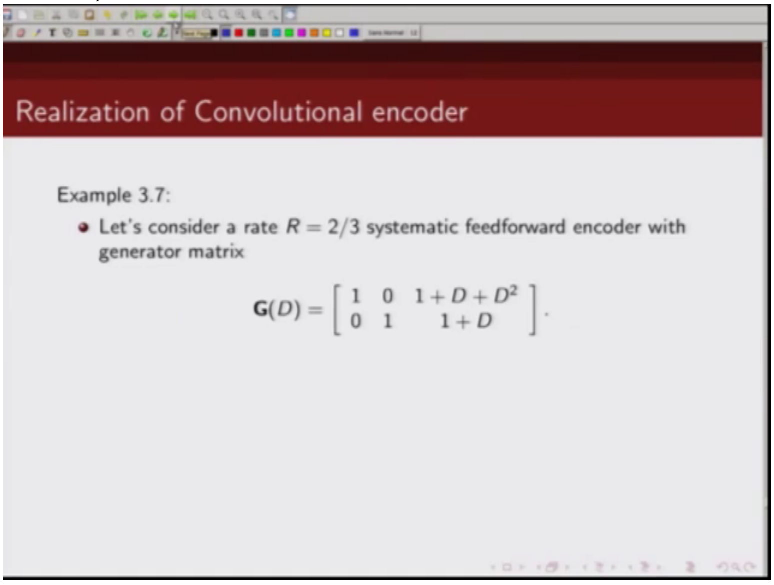
(Refer Slide Time 33:37)



the inputs are directly coming to the adder. So this term is corresponding to f 0 u of D, this term corresponds to f 1 D of u of D. Where as in the controller canonical form the leftmost term was f naught and rightmost was f m. Here just opposite. So you can see this is f 0 term, f 1 term, f 2 term and similarly in the denominator this is q 1, this is q 2, like that this will be q m. Same generator matrix can be realized using 2 different forms.

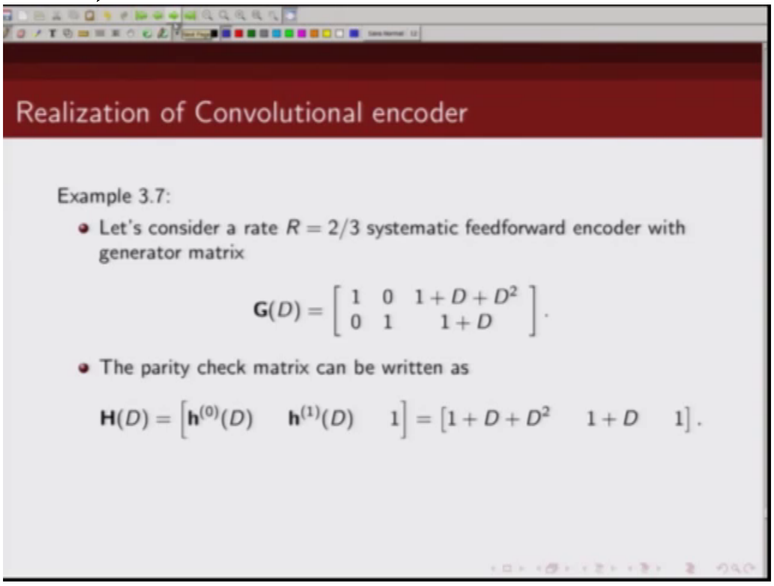
So let us take an

(Refer Slide Time 34:26)



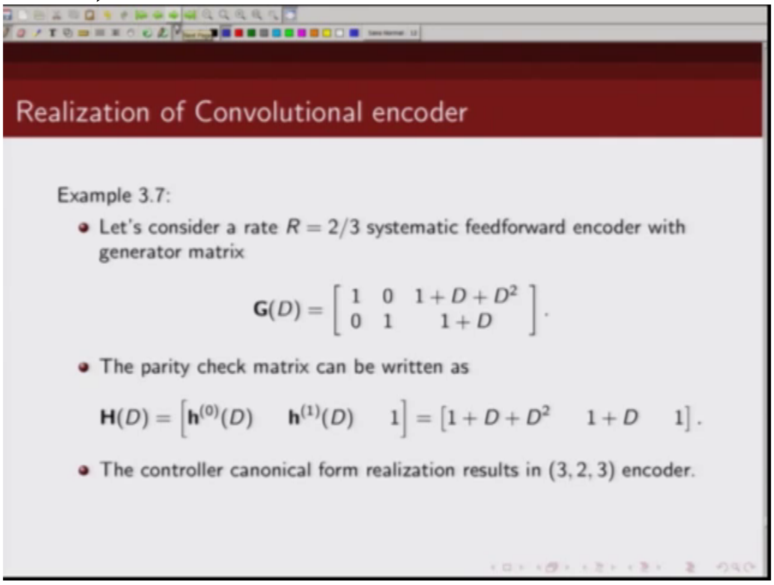
example to illustrate this. So we are considering a rate 2 by 3 systematic feed forward encoder whose generator matrix is given by this. Now let us try to realize this generator matrix using controller canonical form realization and observer form realization. So the parity check matrix for

(Refer Slide Time 34:56)



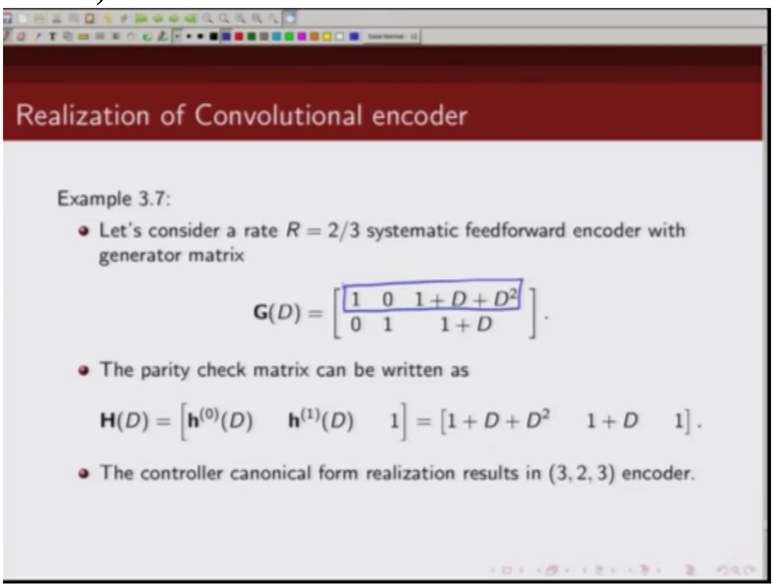
this is given by this expression. We will just

(Refer Slide Time 35:01)



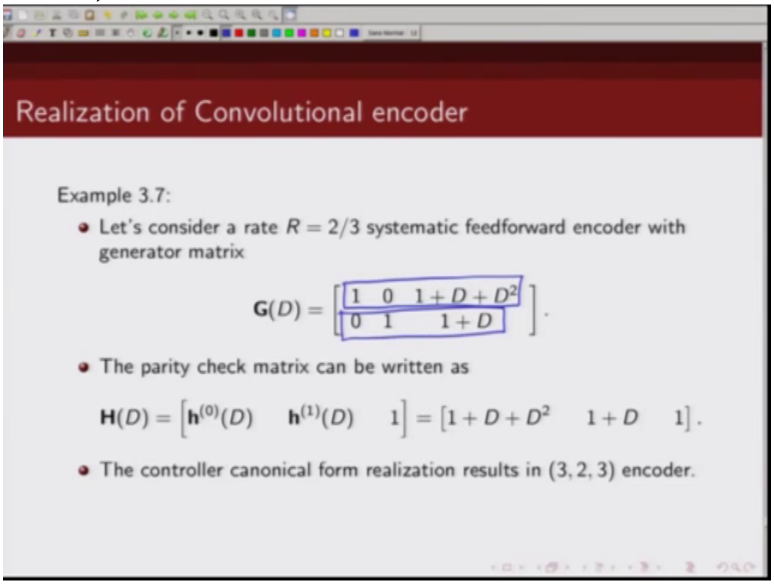
show you that, so in controller canonical form realization, we have, so there are 2 inputs here. So we will have one set of shift registers for each of the input. So we will have one set of shift registers for this

(Refer Slide Time 35:20)



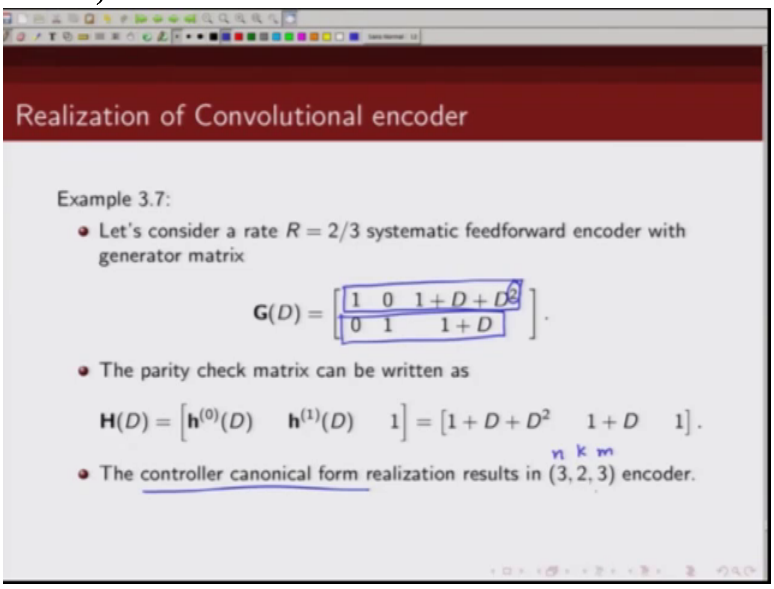
and one set of shift registers for this.

(Refer Slide Time 35:24)



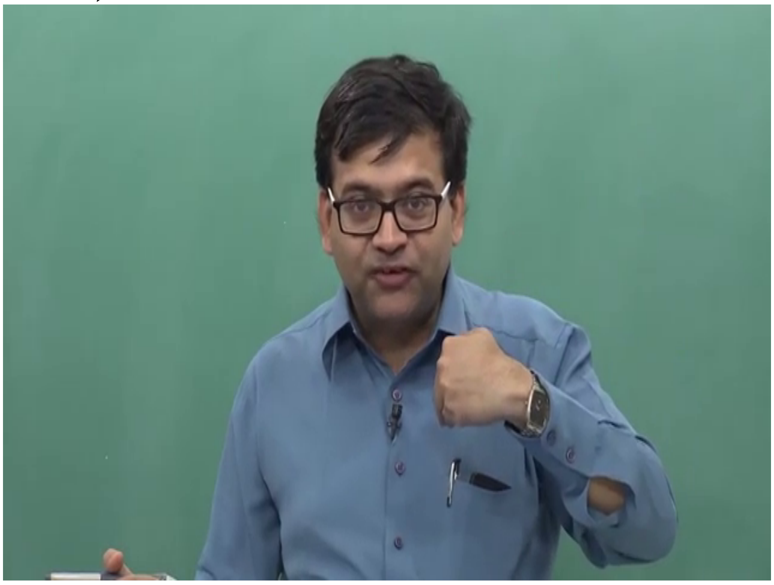
And to realize this we need two memory elements because here, the highest degree of D is 2. And to realize this, we require 1 memory element. So total we would require 3 memory elements. So that's what I said, for controller canonical form realization for this rate two third, this is my n, this is k and this is memory order.

(Refer Slide Time 35:53)



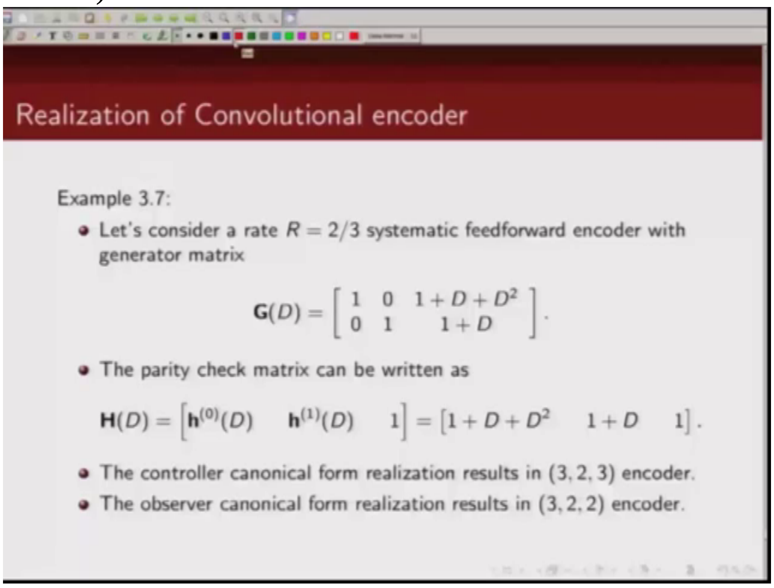
p basically requires 3 memory elements to represent this convolutional encoder in the controller canonical form realization. Now what observer canonical form realization? In observer canonical form realization we use one set of shift registers

(Refer Slide Time 36:15)



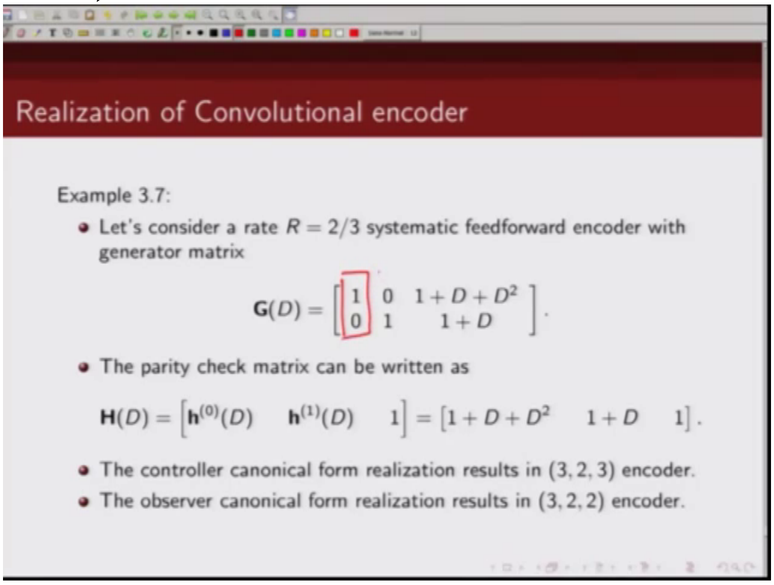
for each of the n coded bits. So how many coded bits we have,

(Refer Slide Time 36:21)



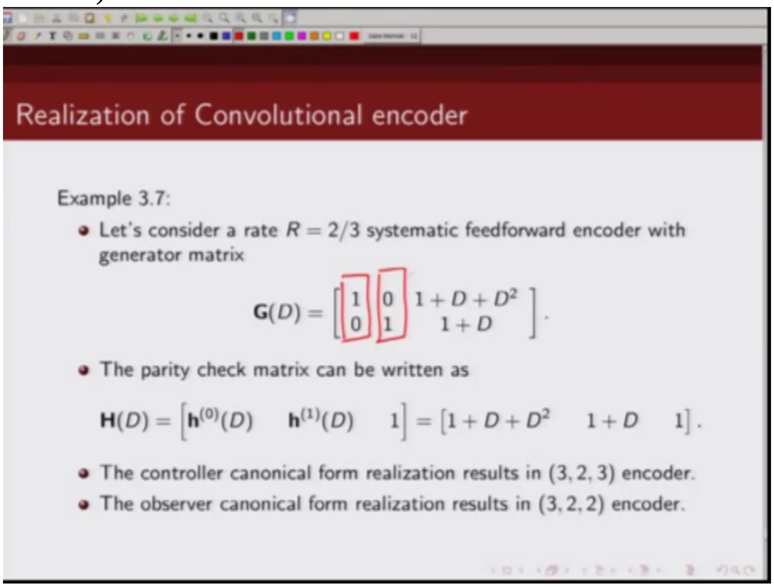
we have 3. One is this, one is this

(Refer Slide Time 36:25)



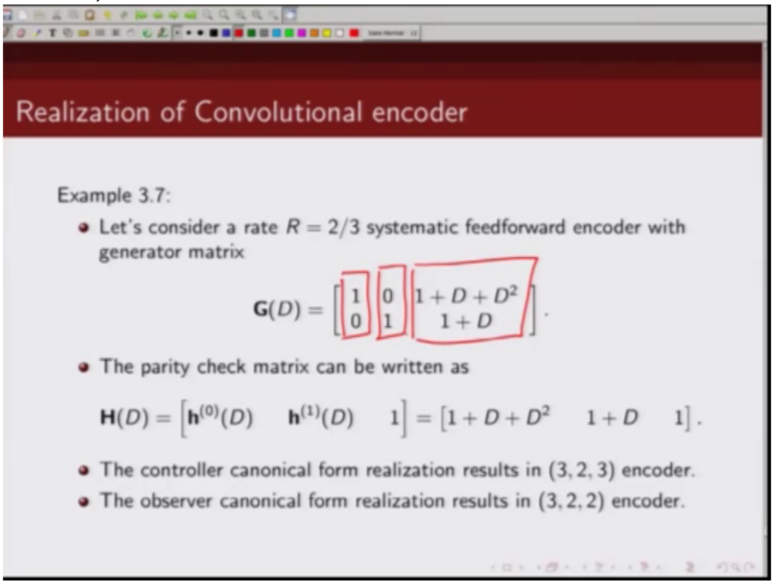
and

(Refer Slide Time 36:27)



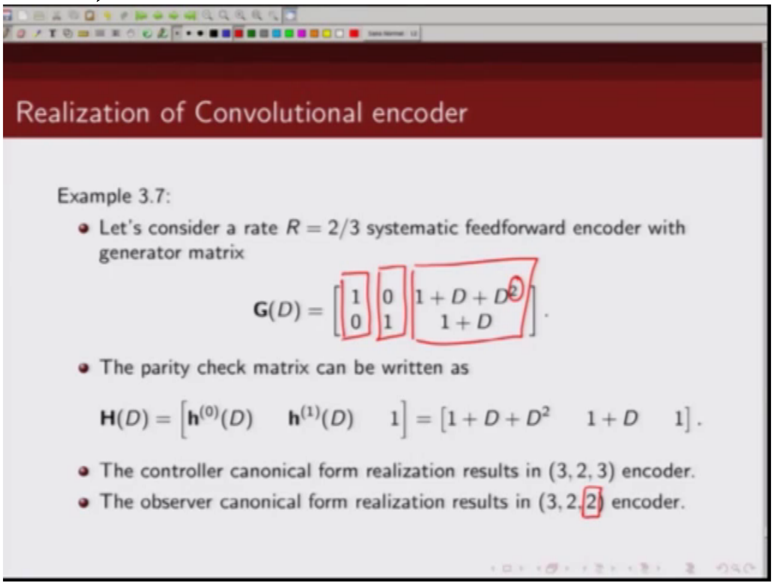
one is this.

(Refer Slide Time 36:30)



Now how many memory elements you require to represent this? Zero, directly the input is coming in here. Here zero, the direct input is coming here and what about this, its maximum degree is 2, we will require 2. So overall for this generator matrix if we try to realize it using observer canonical form realization, we require only 2 memory

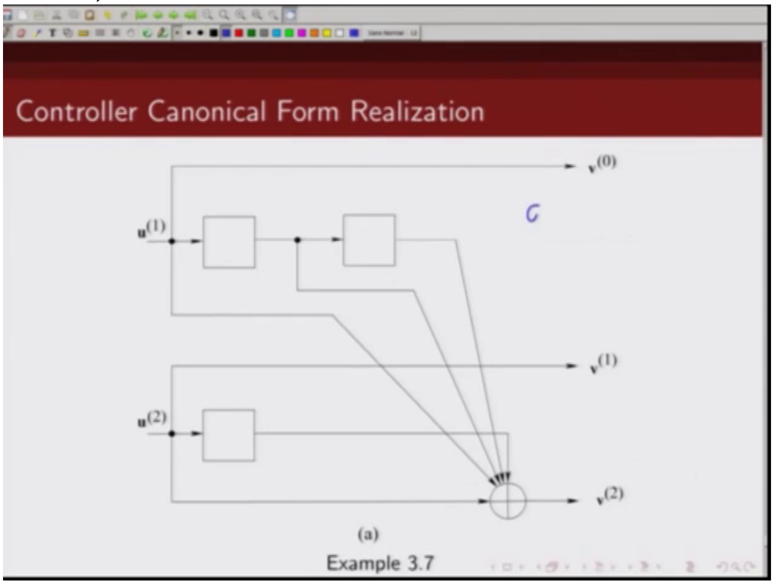
(Refer Slide Time 37:00)



elements.

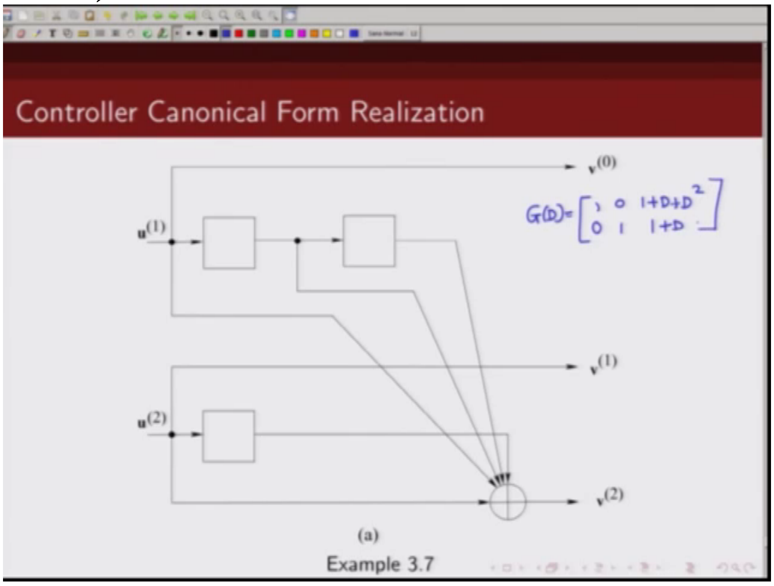
And in the next slide I am going to show you those 2 encoder realization. So let me just write down the generator matrix. My generator matrix G of D

(Refer Slide Time 37:16)



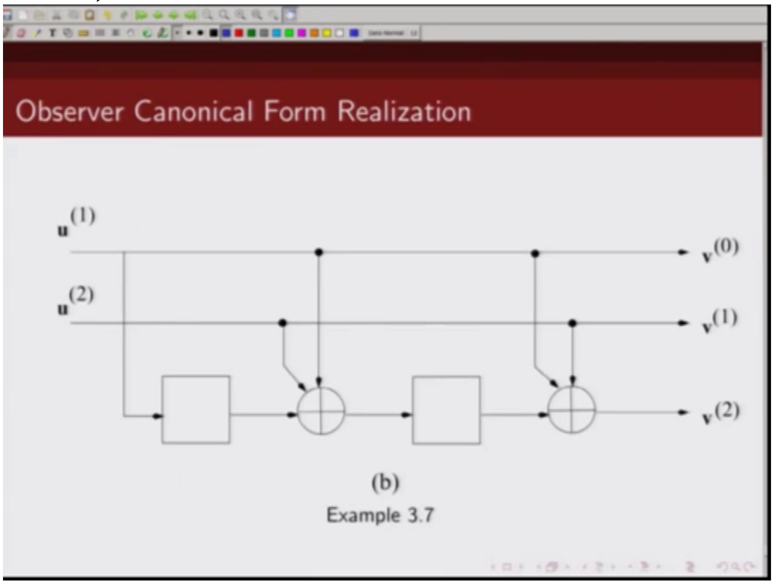
is 1 0 0 1, then what do we have, 1 plus D plus D square, 1 plus D plus D square, 1 plus D,

(Refer Slide Time 37:31)



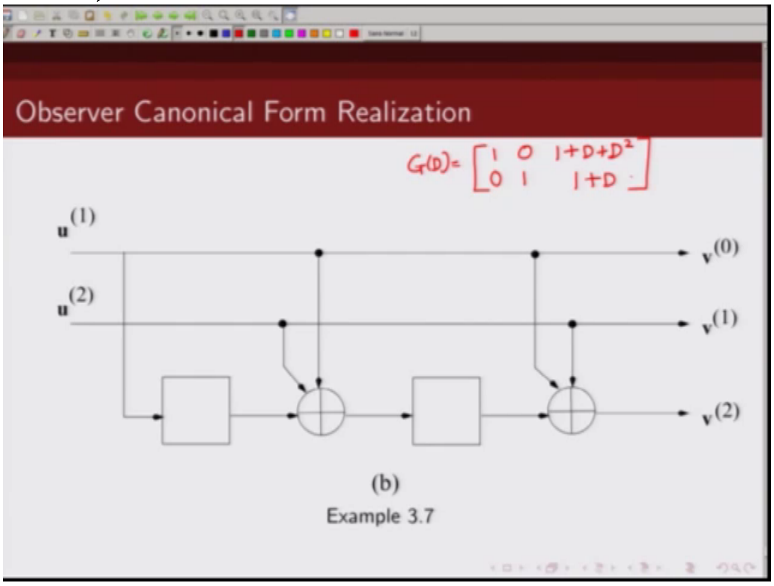
so one set of shift register for this. So this maximum degree is of these 2, so we use 2 memory elements and for this maximum uh degree of d is 1, so we use 1 memory element, Ok. Now what is the first output? First output is directly input u 1, so this is my u 1. Second output is directly u 2, this is basically this and the third output is 1 plus D plus D square of u 1 so this is u, this is the D of u D, u 1 D and this is D square of u 1 D plus 1 plus D times u 2 D. So 1 plus 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 require total 3 memory elements, 1, 2 and 3. Now let us see

(Refer Slide Time 38:55)



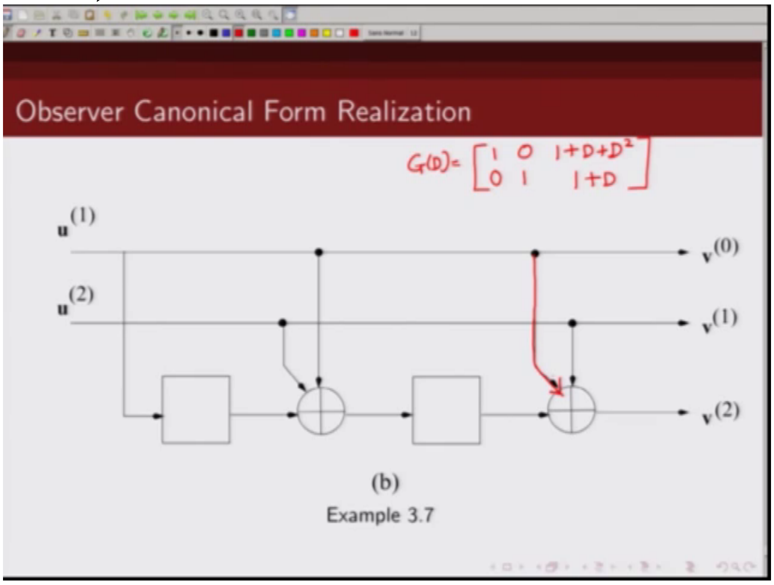
for the observer canonical form realization. Again let me write down my generator matrix. This is 1 0 0 1, 1 plus D plus D square 1 plus D. So we said one set of

(Refer Slide Time 39:16)



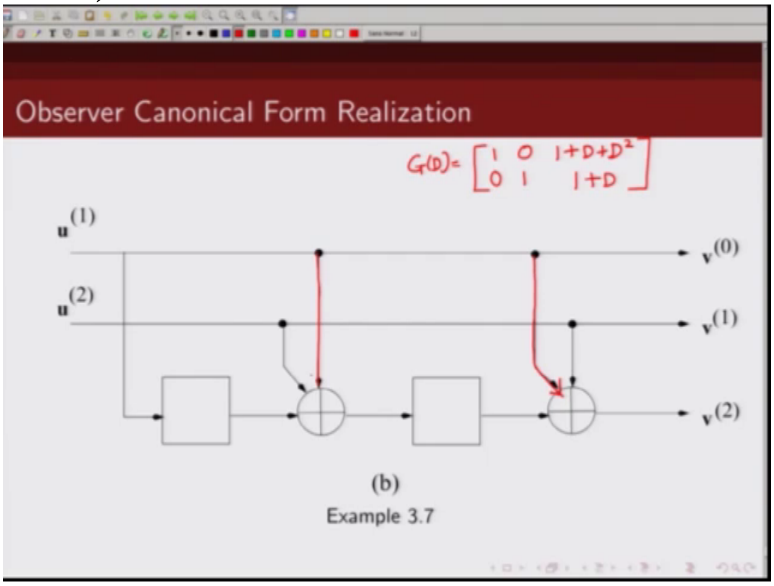
register. What is the maximum delay element here? Zero, so you can see directly. What about this? Again maximum degree of D is basically zero so no shift register. And here for the third line, D is 2 so we took 2 D. So what is the final output then? First one is, first coded bit is just u 1 of D, this is what it is. Second coded bit is u 2 of d, like this. And third coded bit is 1 plus D plus D square of u 1 of D. So what is u 1 of D? u 1 of D is,

(Refer Slide Time 40:15)



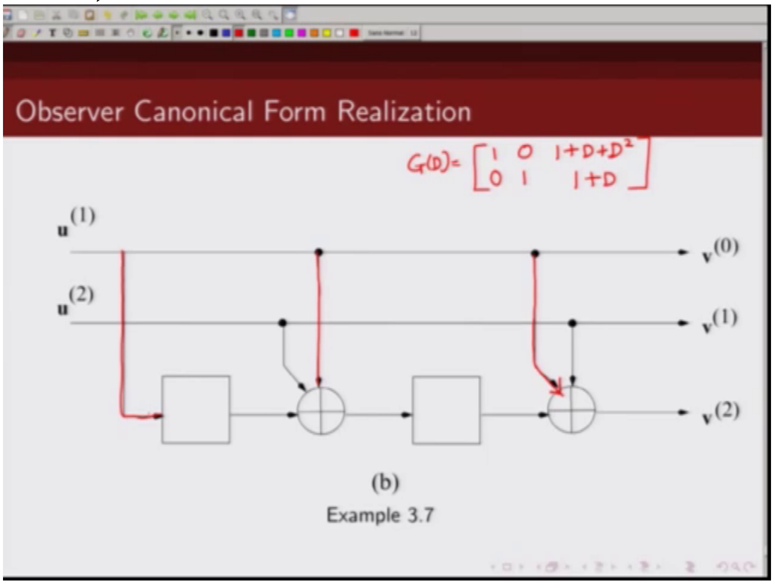
what is D times u 1 of D? That is this term. And what is D square

(Refer Slide Time 40:23)



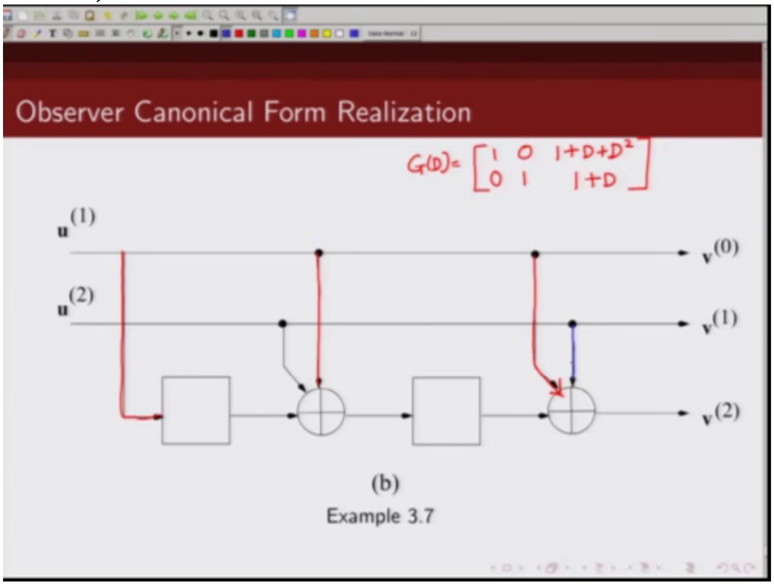
what is D times u 1 of D? That is this term. And what is D square of u 1 of D? That is this term,

(Refer Slide Time 40:29)



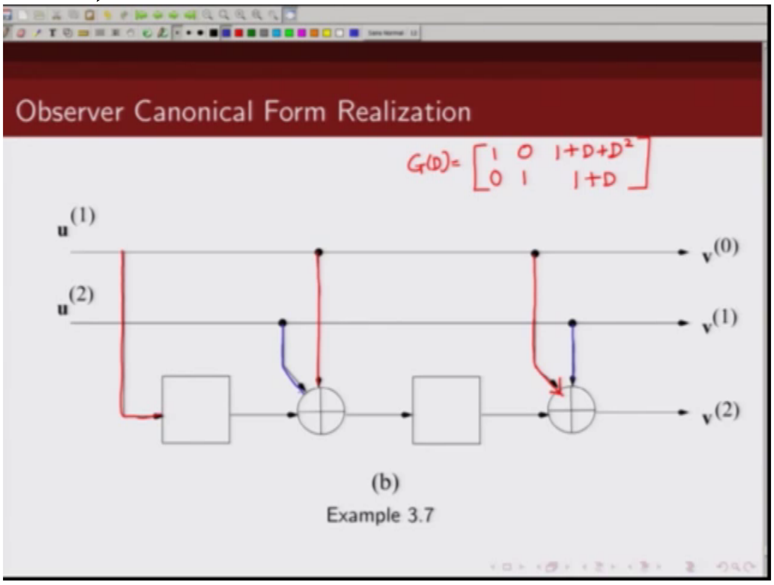
fine plus 1 plus D times u 2 of D. So then what we have is u 2 of D is this and

(Refer Slide Time 40:43)



D times u 2 of D is this.

(Refer Slide Time 40:46)

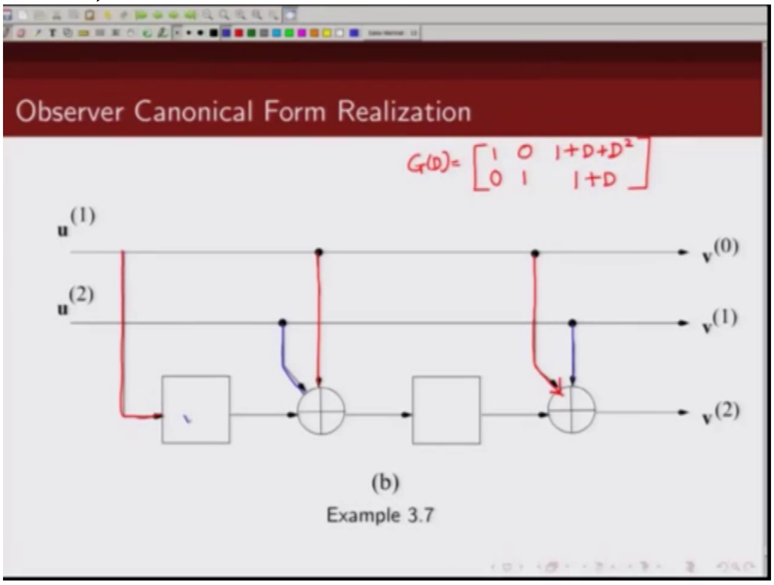


So this is our observer canonical form realization for this convolutional encoder with this generator

(Refer Slide Time 40:57)

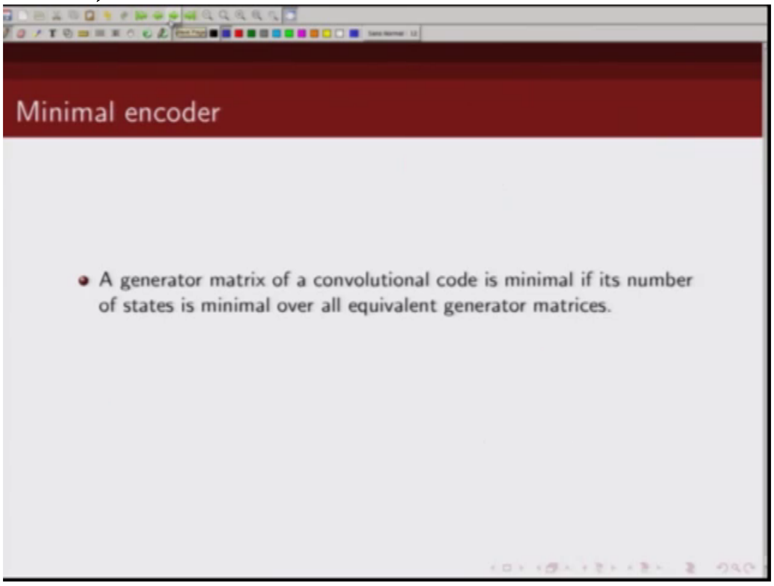
matrix and note we only require two, 1, 2

(Refer Slide Time 41:01)



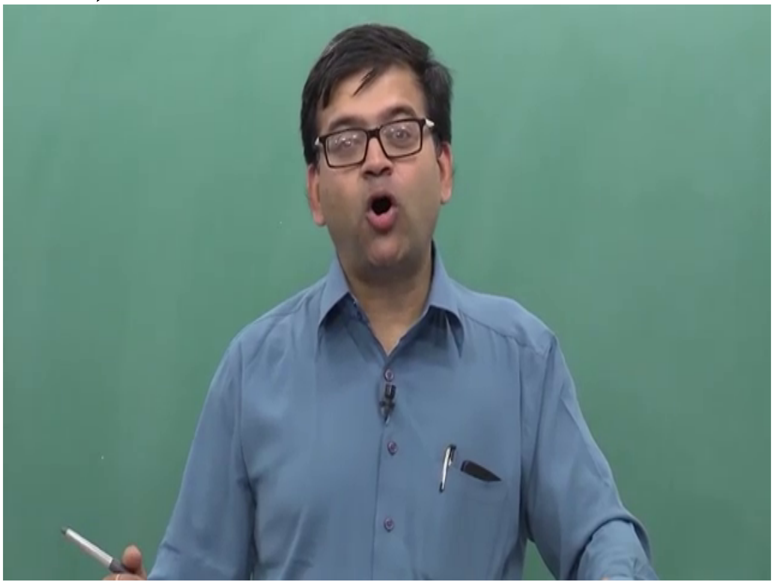
we required two memory elements. So same encoder here with two memory elements, for the controller canonical form realization we require 3 memory elements. So that brings

(Refer Slide Time 41:17)



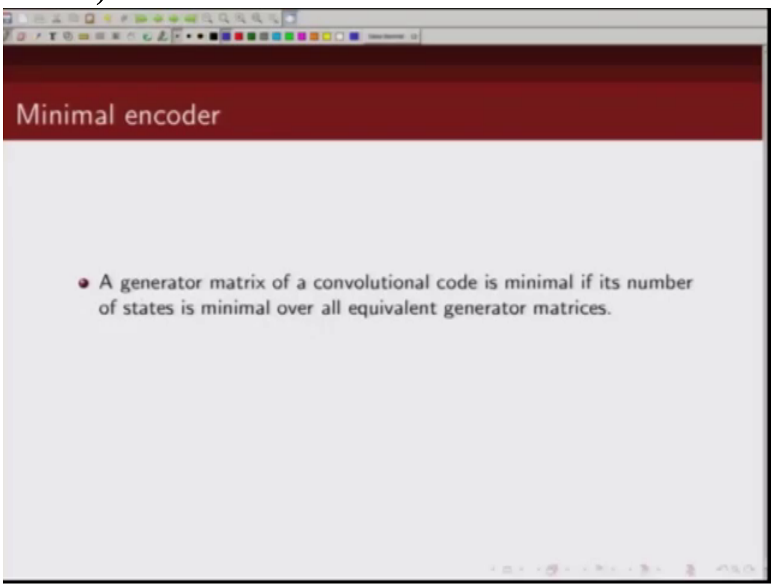
us to this notion of

(Refer Slide Time 41:20)



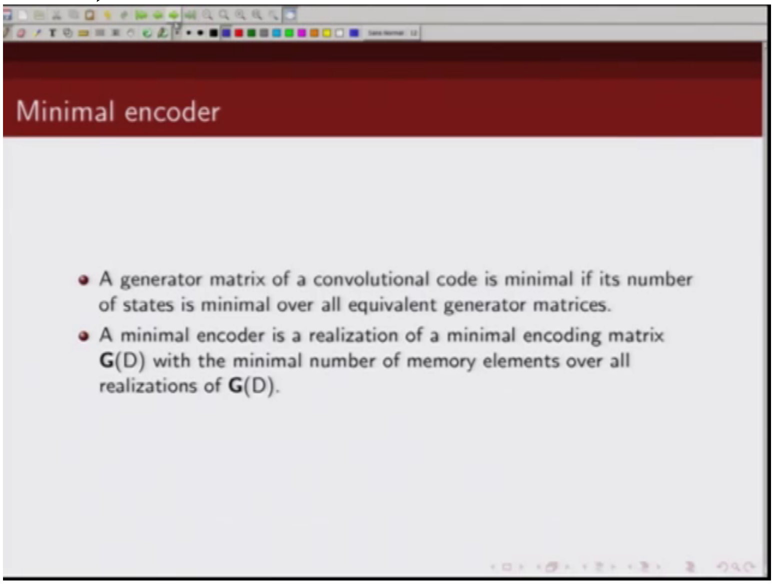
minimal encoder. We saw the same encoder; convolutional encoder with same generator matrix can be realized using 2 different ways, one that resulted in 3 memory elements, other that resulted in 2 memory elements. So we say a generator

(Refer Slide Time 41:36)



matrix is minimal if the, if its number of states is minimal over all possible equivalent generator matrix and among

(Refer Slide Time 41:47)

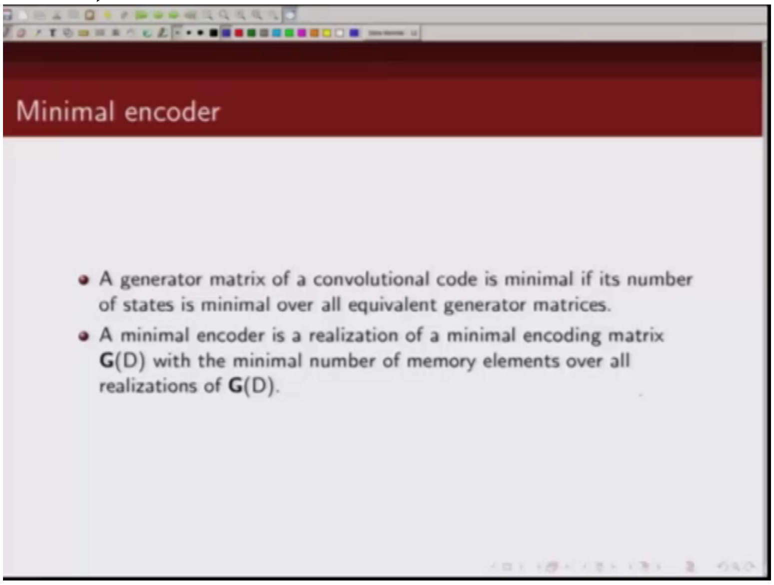


the minimal encoder matrix, a minimal encoder is basically a realization of a minimal encoding matrix which will result in minimum number of memory elements used to represent that particular convolutional encoder. So we define a minimal encoder as

(Refer Slide Time 42:11)

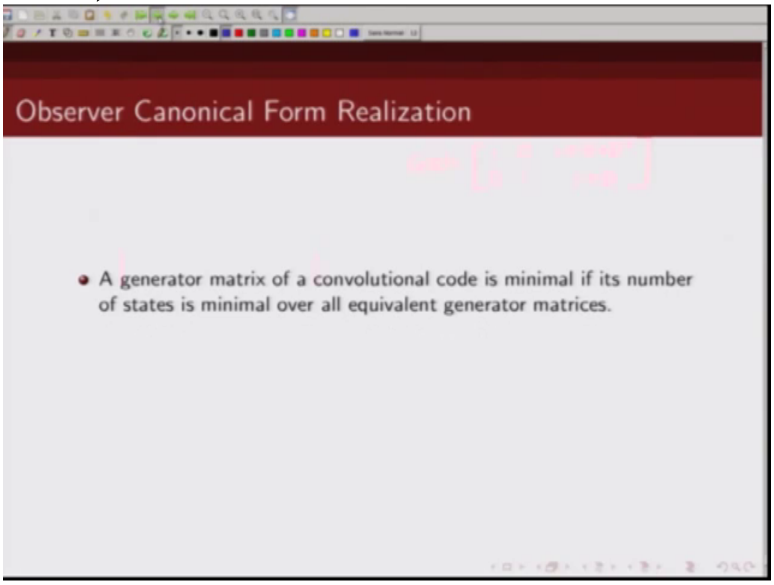
the minimal

(Refer Slide Time 42:14)



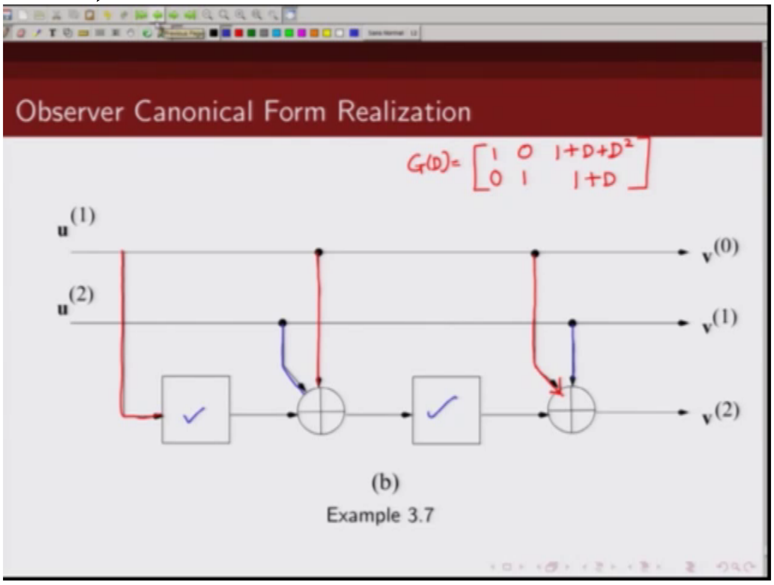
realization of a minimal encoding matrix. So the minimal encoder should result in minimum number of memory elements used to represent that particular convolutional encoder and for the example we have considered,

(Refer Slide Time 42:31)



this, in this case you can see

(Refer Slide Time 42:34)



from the generator matrix the maximum degree of d is 2. So we at least need 2 memory elements to represent it and you can see the observer canonical form realization in this particular example will result in minimal encoder of this convolutional encoder. So this realization will result in minimal encoder realization for this convolutional encoder, thank you.

(Refer Slide Time 43:01)