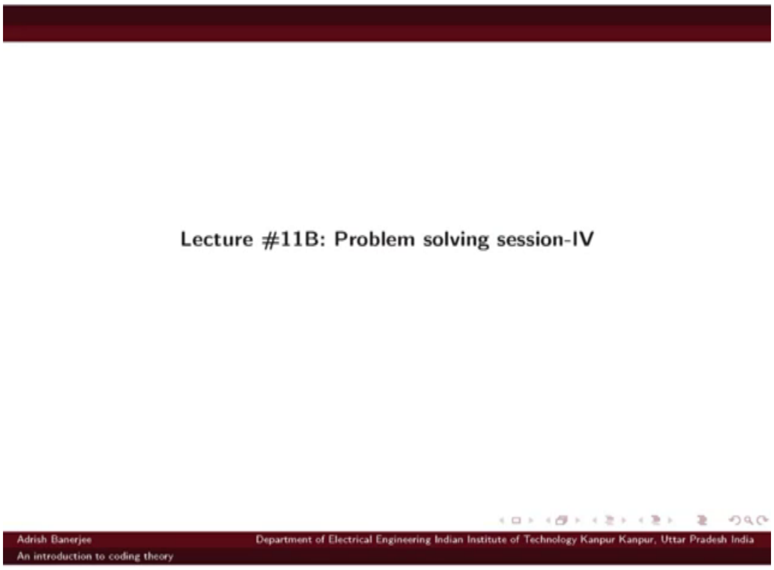
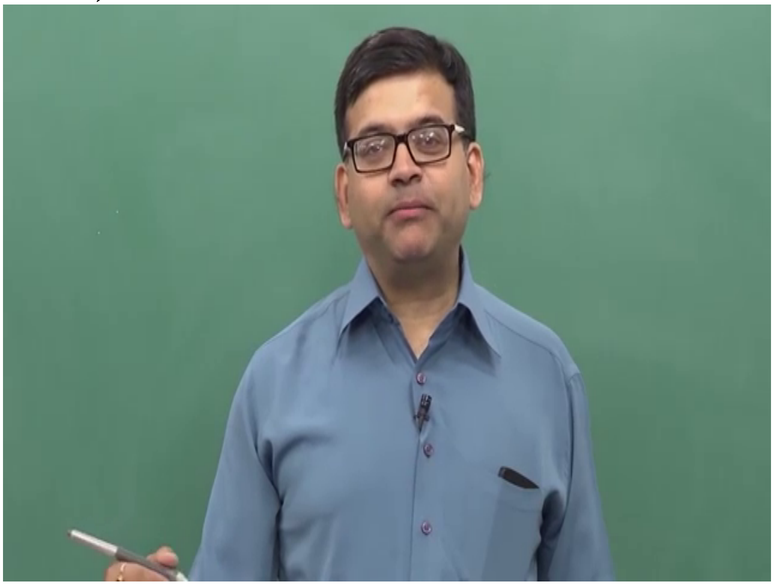
An Introduction to Coding Theory
Professor Adrish Banerji
Department of Electrical Engineering
Indian Institute of Technology, Kanpur
Module 05
Lecture Number 21
Problem Solving Session-IV

(Refer Slide Time 00:14)



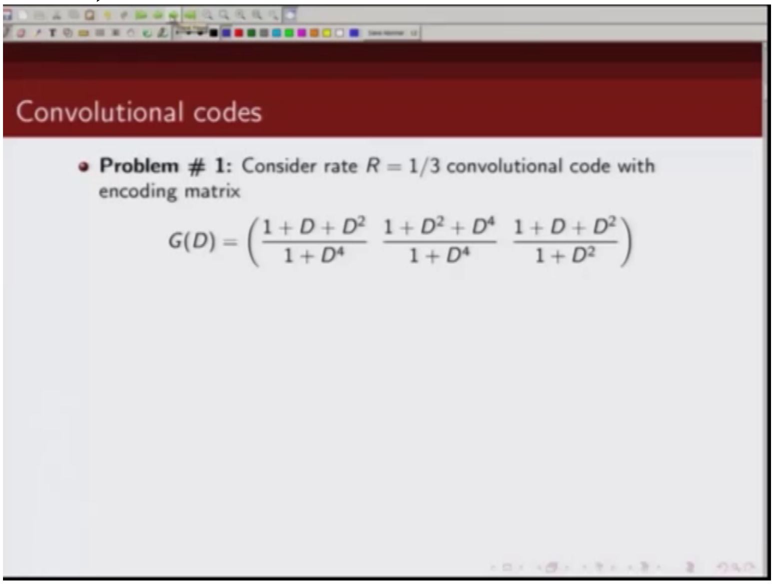
So before we go to concatenated codes, let us spend

(Refer Slide Time 00:18)



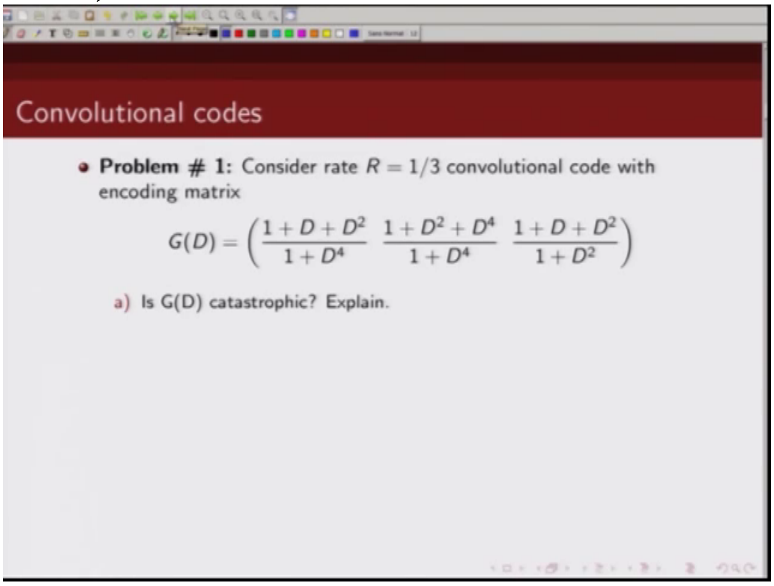
some time solving some problems.

(Refer Slide Time 00:21)



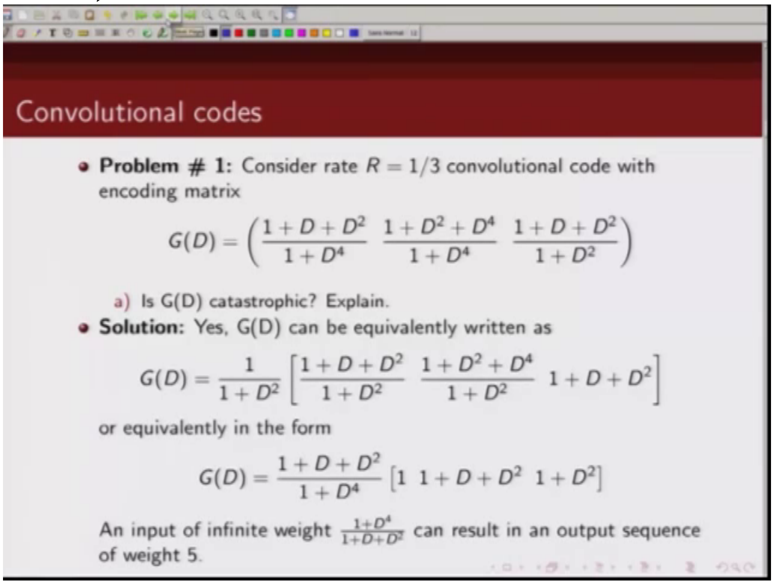
So the first question is you are given a rate one third convolutional code with generator matrix G of D which is given by this.

(Refer Slide Time 00:33)



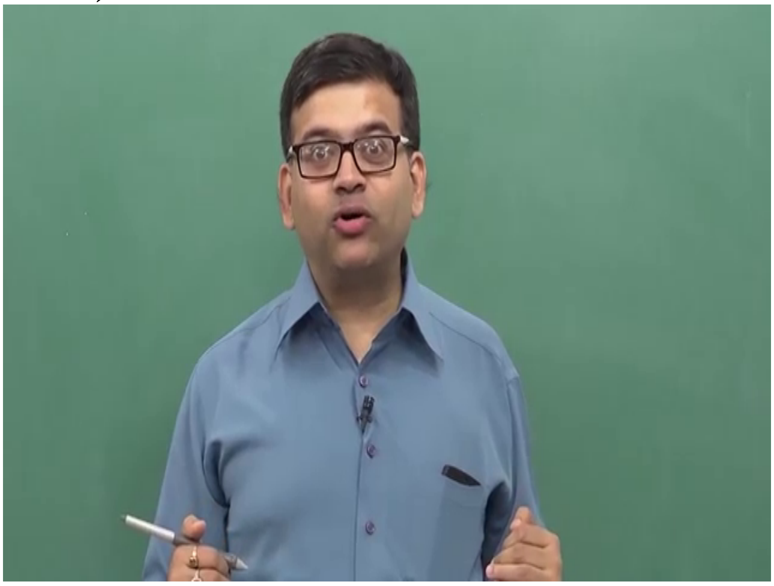
The first question is, is this a catastrophic encoder? Will an encoder which has a generator matrix like this; will this result in a catastrophic encoder? So if

(Refer Slide Time 00:50)



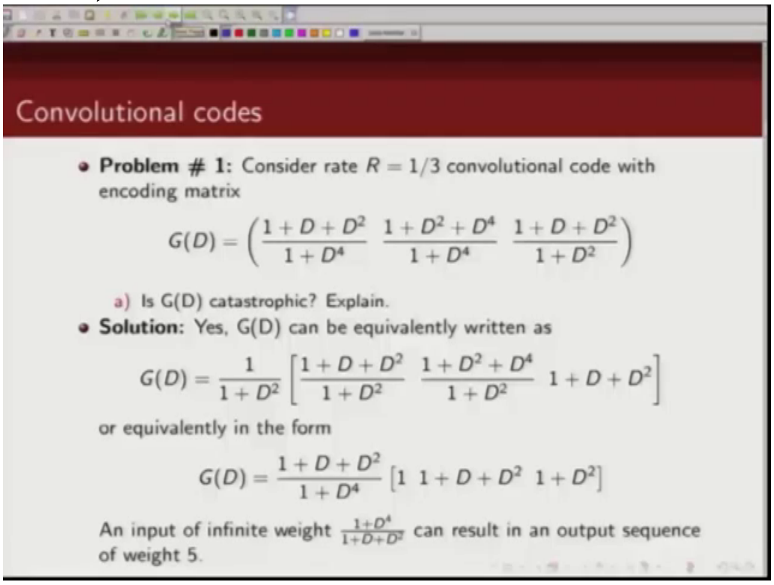
you recall what is a catastrophic encoder, a catastrophic

(Refer Slide Time 00:54)



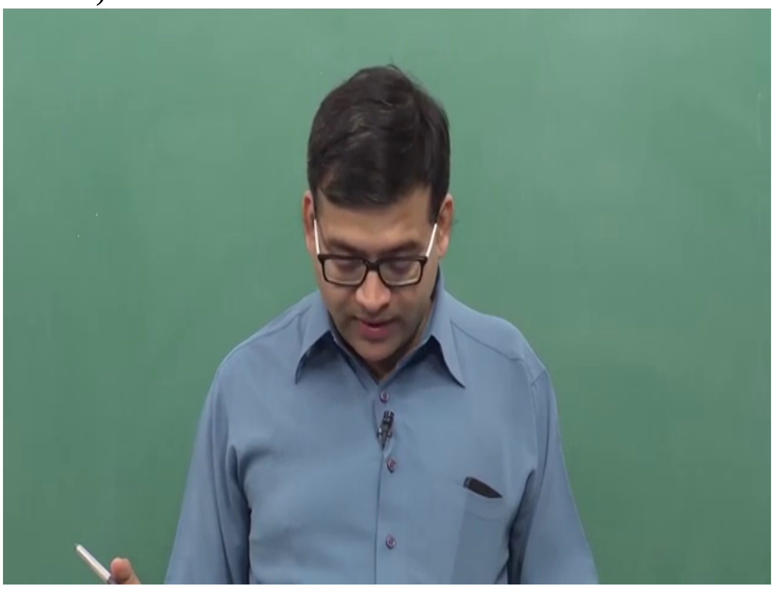
encoder generates a finite weight output corresponding to an infinite weight input sequence. Now if we try to look it in terms of state diagram, in a state away from all zero state, there is a self loop around a state where a non-zero input results in all zero output, right. Now let's look at this generator matrix

(Refer Slide Time 01:23)



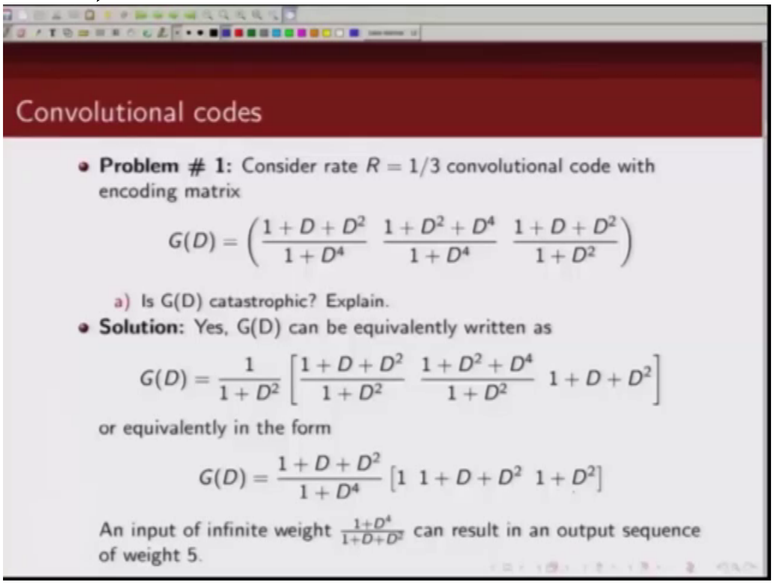
and let us try to simplify, put it in a minimal form. So we can see the denominator, one plus D square is common. So if we take that out, we get here 1 plus D plus D square and this is 1 plus D square, this is 1 plus D square plus D four divided by 1 plus D square and this is 1 plus D plus D square. Similarly we see in the numerator there is a common term 1 plus D plus D square. If we take that out, what we get here is then this is 1, 1 plus D plus D square and 1 plus D square. Now how do we know whether this will result in a catastrophic

(Refer Slide Time 02:07)



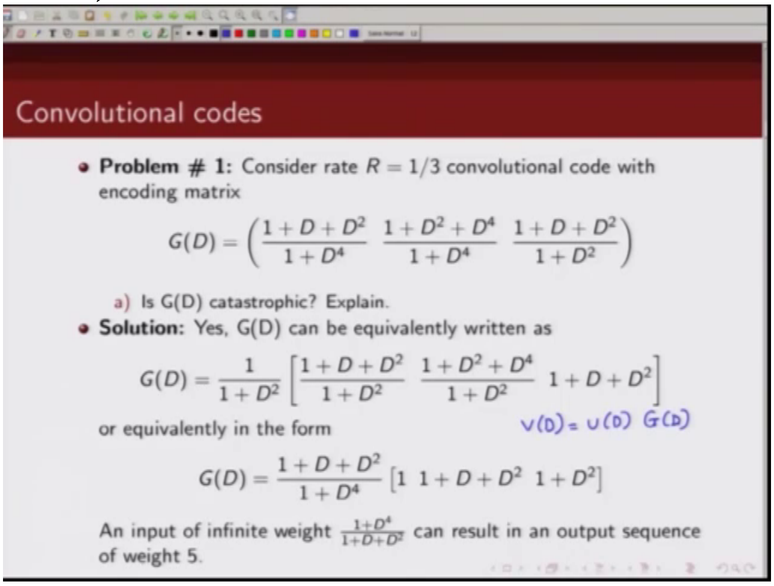
encoder or not. So look at

(Refer Slide Time 02:10)



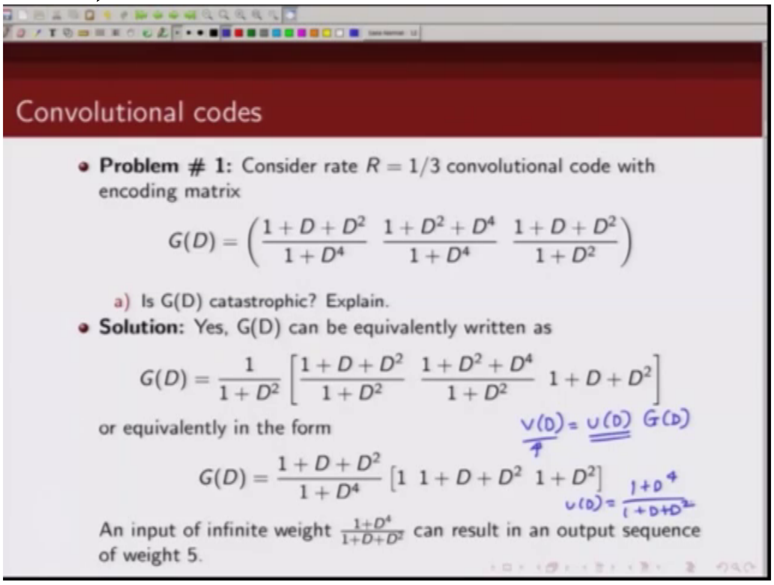
this particular generator matrix. Now what is my output sequence? My output sequence v D is u D times g D.

(Refer Slide Time 02:21)



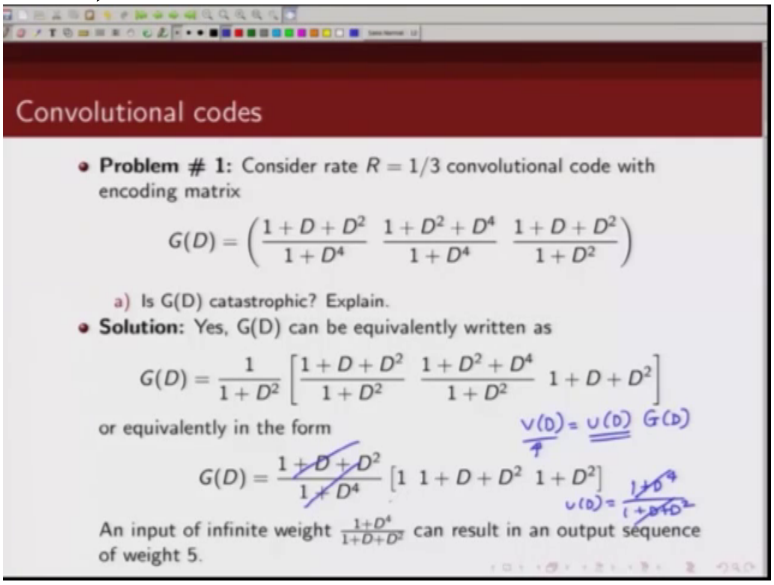
Now is there any input sequence which is of infinite weight but can result in a finite output weight for v D? If you pay close attention to g of D we notice that if our input u of D is chosen as 1 plus D four 1 plus D plus D square,

(Refer Slide Time 02:49)



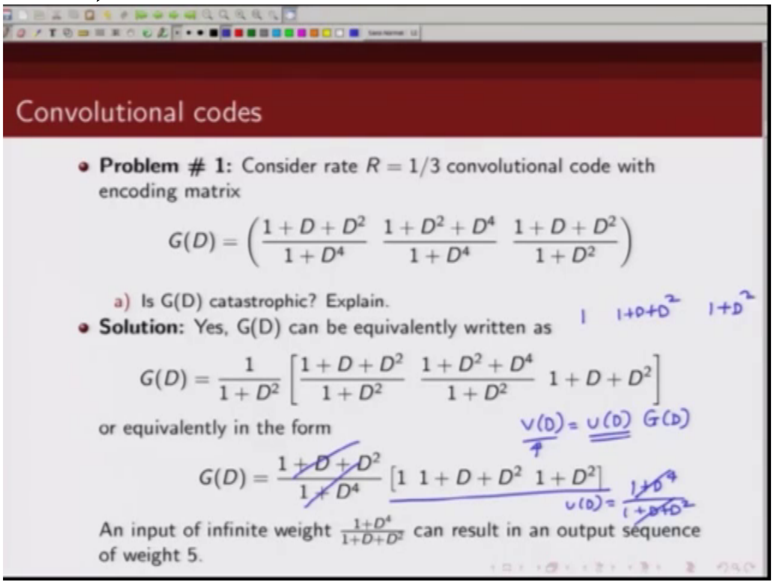
if our input is chosen in this particular fashion, then what would be the corresponding output v D? If the input is chosen this way then output will be u D times g D so this term will cancel this term so what

(Refer Slide Time 03:07)



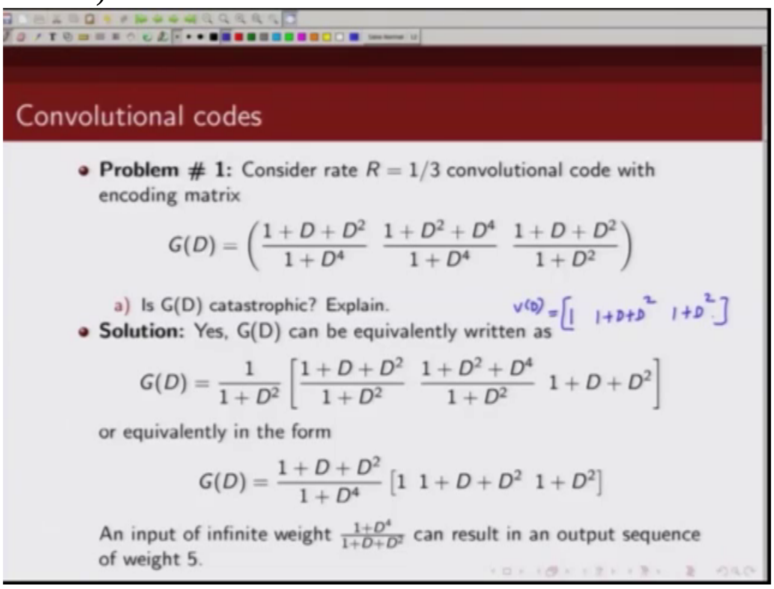
you will be left with is this. So your v of S would be 1, 1 plus D plus D square and 1 plus D square. And what

(Refer Slide Time 03:18)



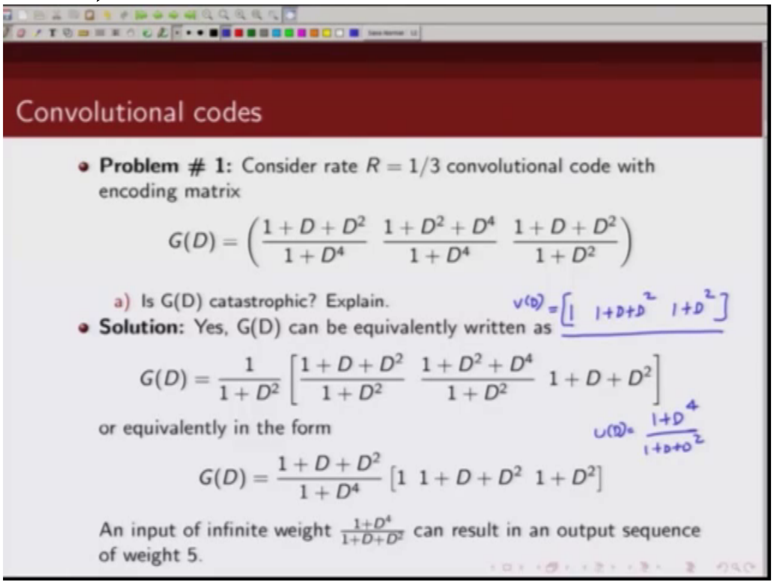
is the weight of this? (Delete kar diya, woh ganda)

(Refer Slide Time 03:21)



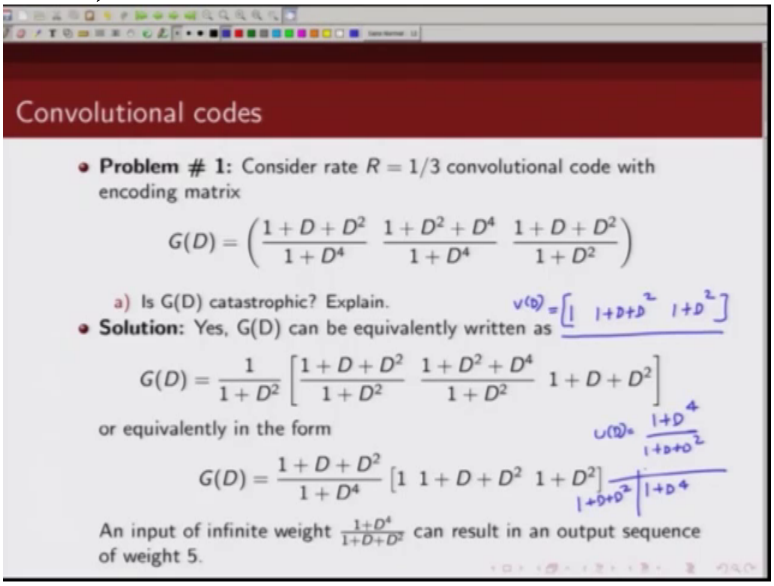
So note here, so the input that will cause this output is given by 1 plus D four by 1 plus D plus D square,

(Refer Slide Time 03:42)



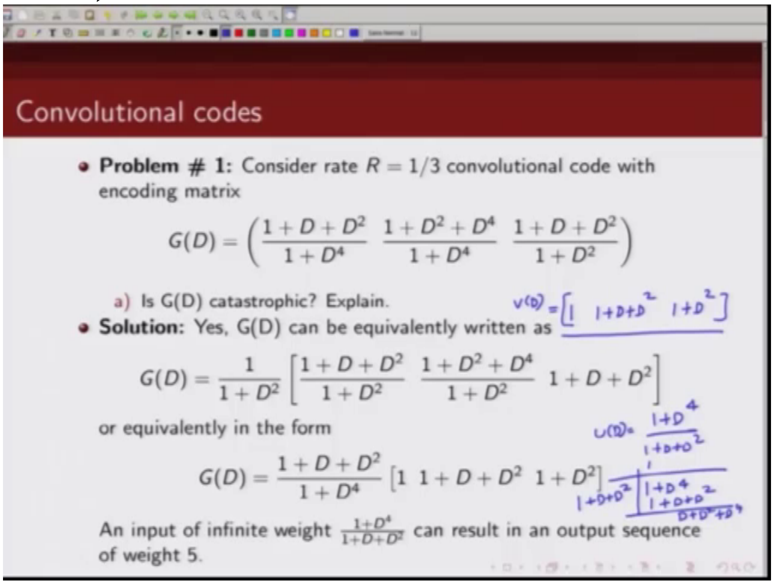
right? Now we can expand this. So let's say 1 plus D 4, this is 1 plus D plus D square. So let's just take

(Refer Slide Time 03:56)



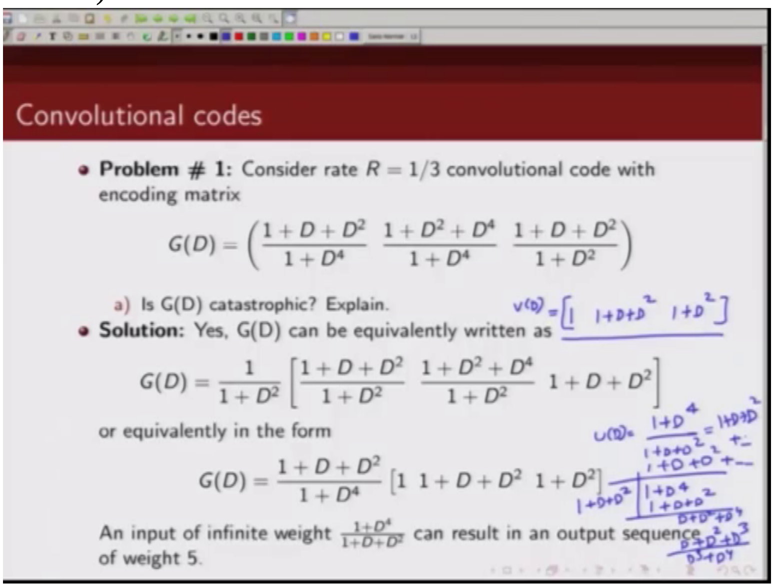
1. 1 plus D plus D square, this will be D plus D square plus D 4, now

(Refer Slide Time 04:05)



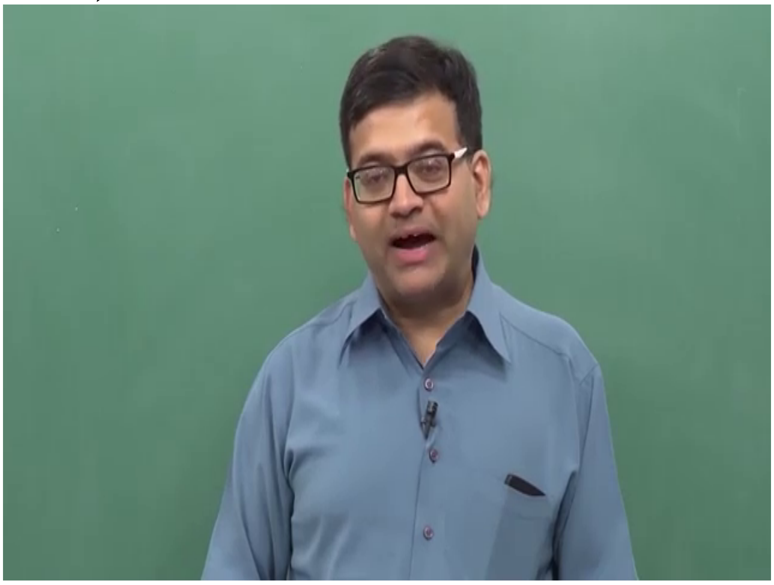
this will be plus D, this will be D plus D square plus D cube, then this will be D cube plus D 4, you can write D square, so like that basically we, we can see that this is a infinite series. The input is an infinite series, 1 plus D plus D square is essentially an infinite series. We can

(Refer Slide Time 04:36)



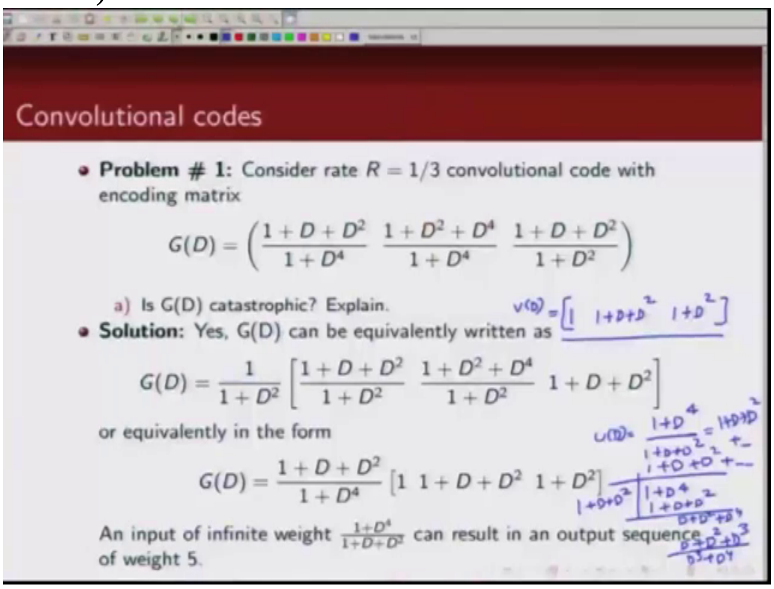
expand it like that where as output is a finite series, it is just 1, 1 plus D plus D square, and the third bit is 1 plus D square. So you can see, input has lots of 1's in it but the output has finite 1s. So this is a case of catastrophic

(Refer Slide Time 04:59)

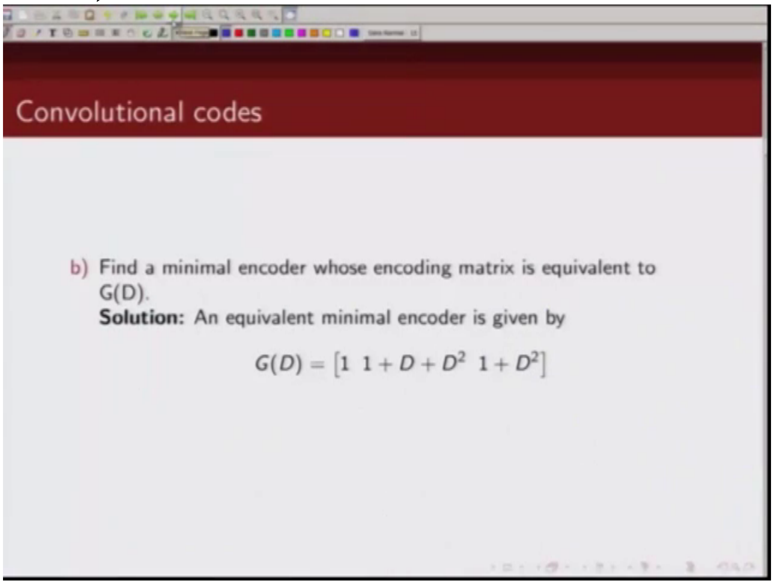


encoder.

(Refer Slide Time 05:00)

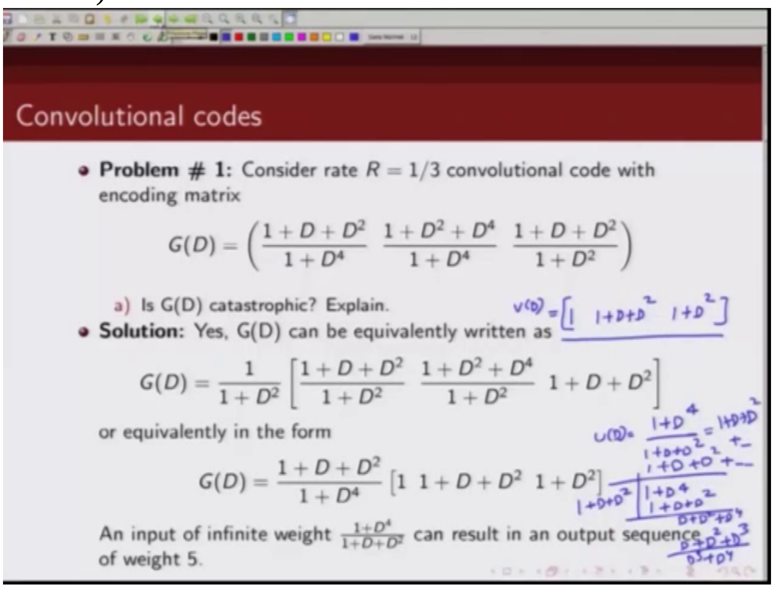


(Refer Slide Time 05:01)



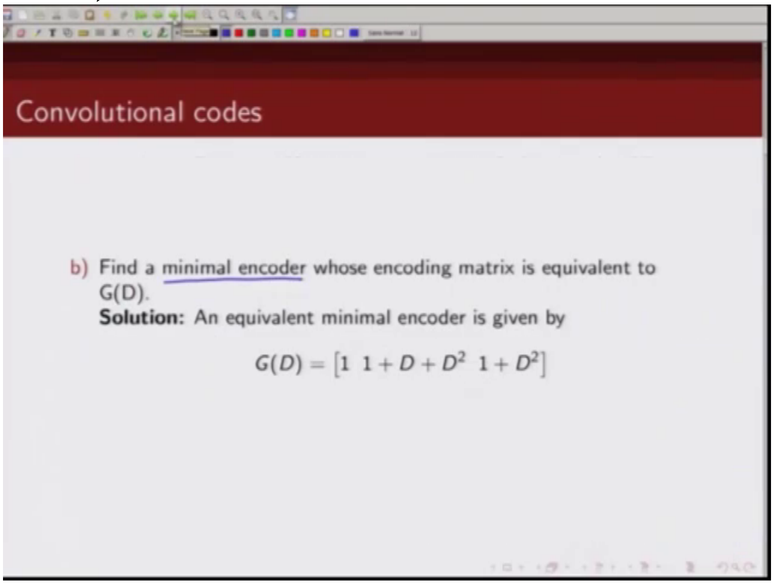
Now the second question is what would be the minimal encoding matrix for the generator matrix given in the previous example.

(Refer Slide Time 05:10)



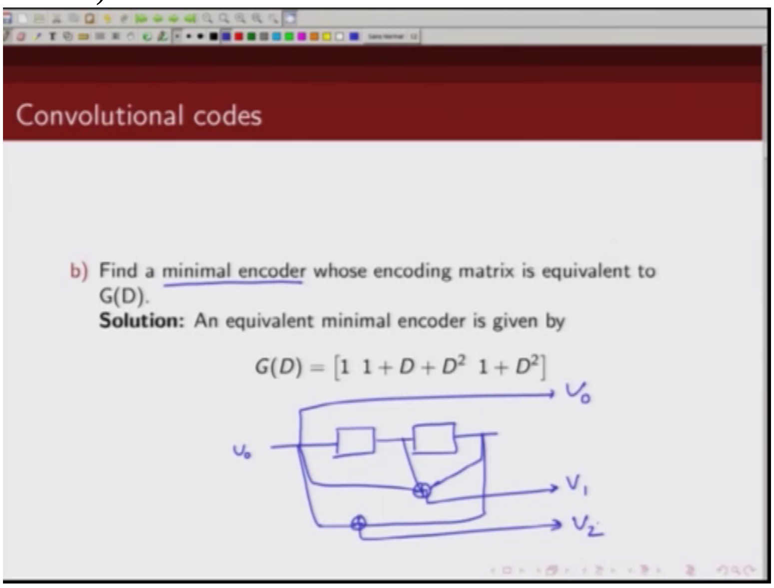
If I ask you, find out the minimal encoding matrix for this encoder. So what do we do? We take out all the common factors. If we take out common factors, then we basically what we get is like this is our minimal encoding

(Refer Slide Time 05:31)



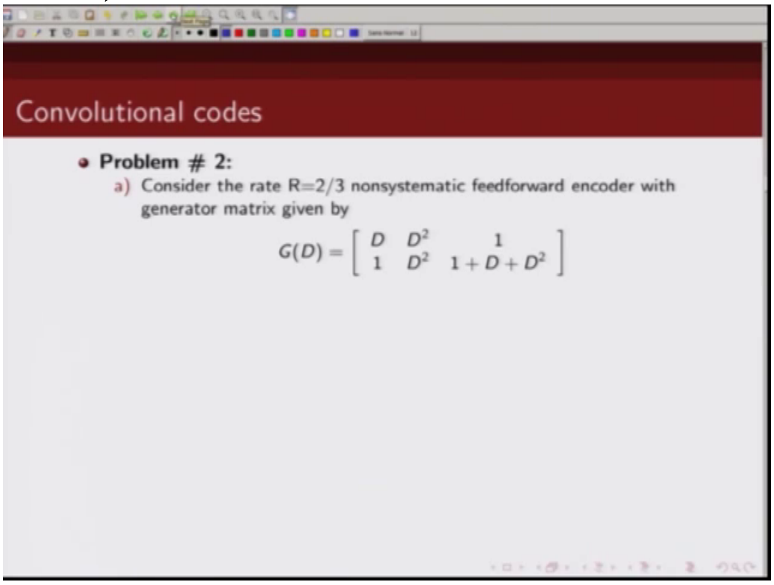
matrix and if we can write, if I ask you to draw this encoder, we can, this is, k is 1, n is 3, the maximum memory is 2 so I am drawing 2 memory elements here. The first coded bit is just 1, so this is the information sequence that goes in. Second one is 1 plus D plus D square. So that's your, let's call it v 0, this is v 1, this is u 0, and the third bit, coded bit is 1, and D square this is your v 2,

(Refer Slide Time 06:22)



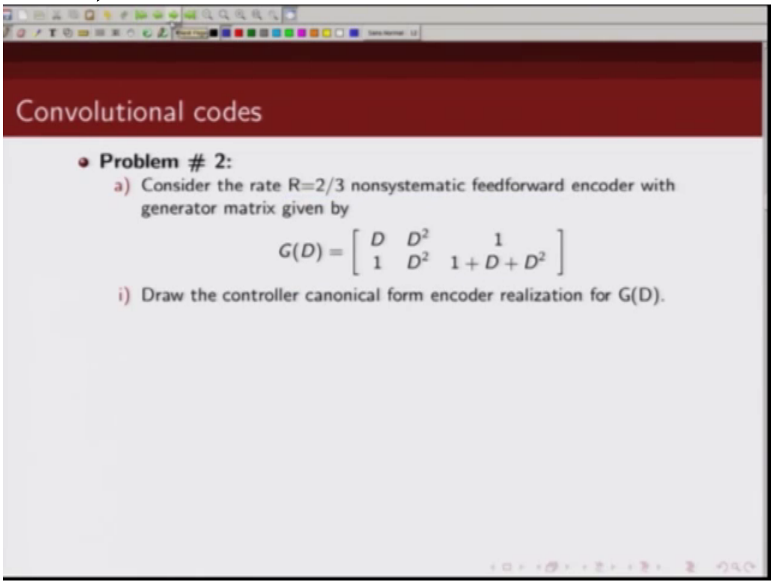
Ok. So this is the minimal encoder for the same generator matrix given in the previous example.

(Refer Slide Time 06:31)



So consider a rate two third non systematic feed forward encoder. So this is a generator matrix for a non systematic code, rate two third and it is a feed forward encoder. There are no feedback polynomials here.

(Refer Slide Time 06:50)

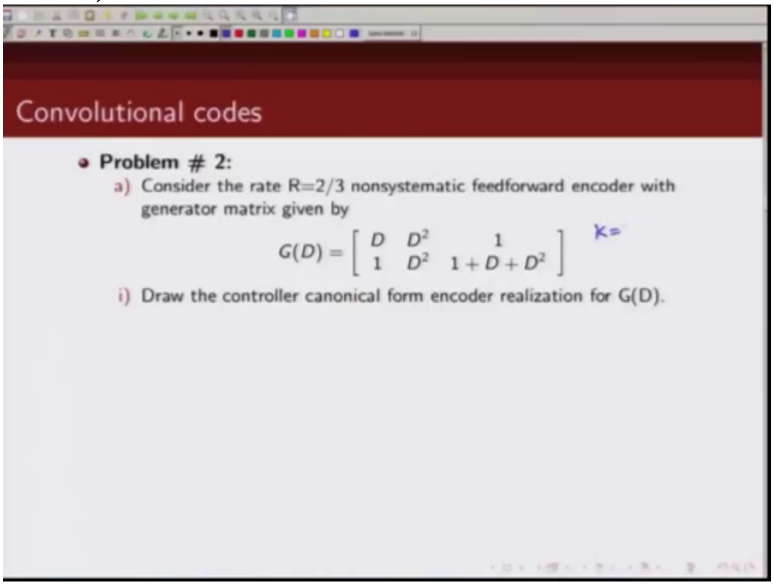


The first question is draw the controller canonical form realization for this generator matrix. Now in controller canonical form realization we have 1 set

(Refer Slide Time 07:05)

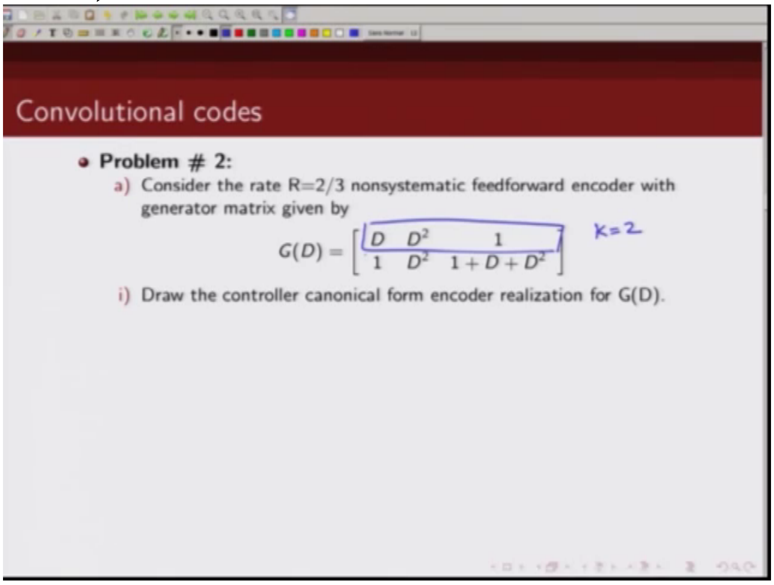
of shift register for input. Now how many inputs do we have here? k is 2

(Refer Slide Time 07:14)



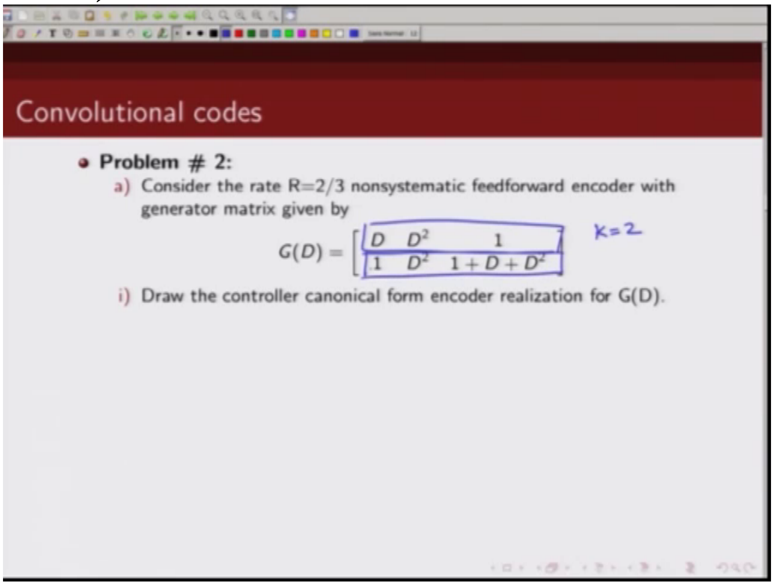
so we will have 2 sets of shift registers for this. One for this and second

(Refer Slide Time 07:24)



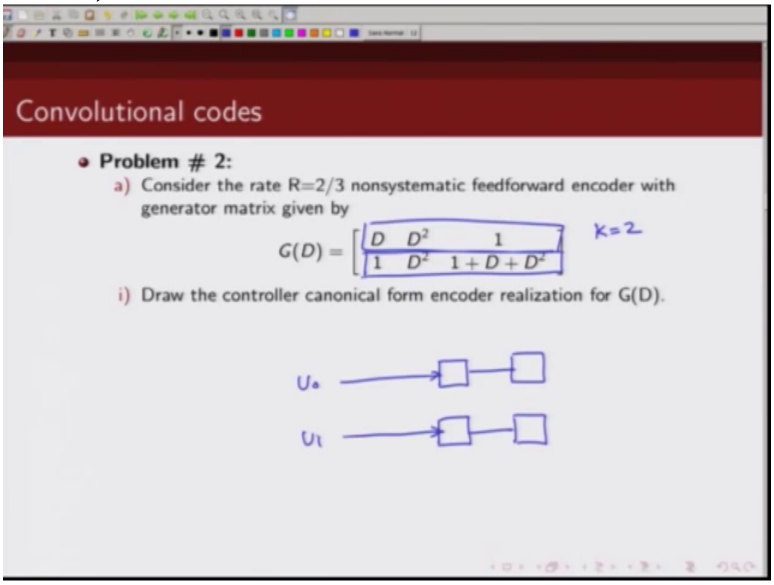
set of shift registers for this. Now what

(Refer Slide Time 07:27)



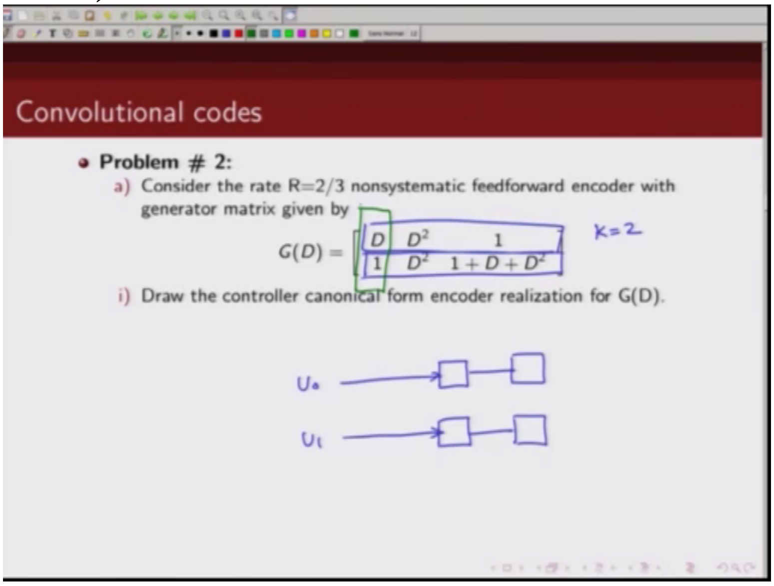
should be the maximum memory for each of these shift registers? You can see here the maximum power of d is 2. So we should have 2 memory elements for the first input. Similarly for the second input also we should have 2 memory elements. Let's call it u 0 and u 1.

(Refer Slide Time 07:54)



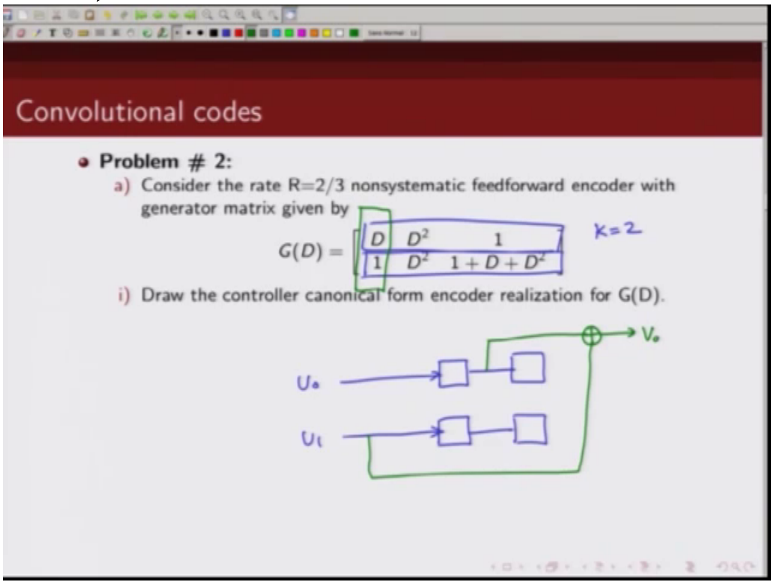
Now there are 3 outputs. So the outputs are, this is 1 output, D times

(Refer Slide Time 08:06)



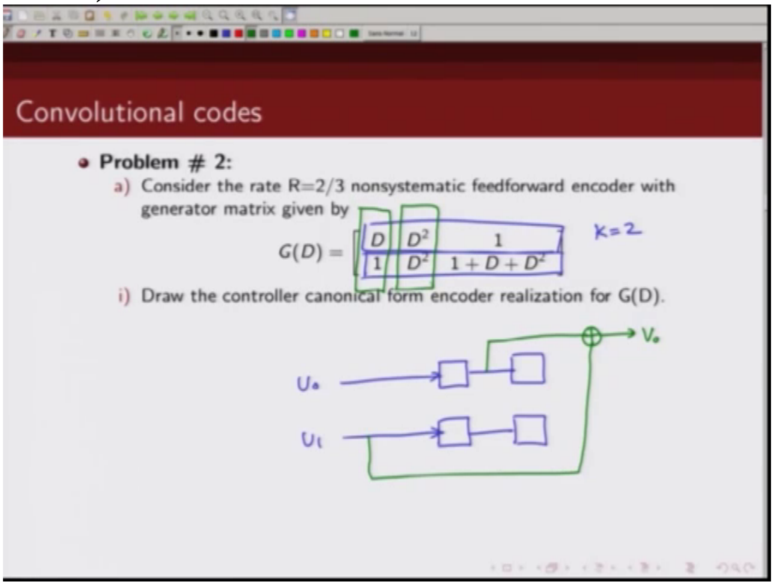
the first and one times u 1, so D times the first input is this, and one times second input so that is this. So this is your first coded bit. Let's call it v 0.

(Refer Slide Time 08:28)



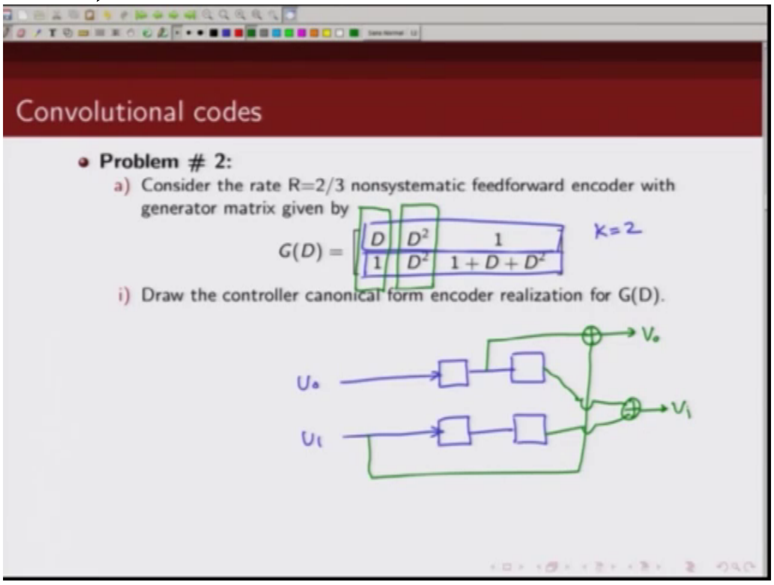
Now what's the second coded bit? This is this term,

(Refer Slide Time 08:33)



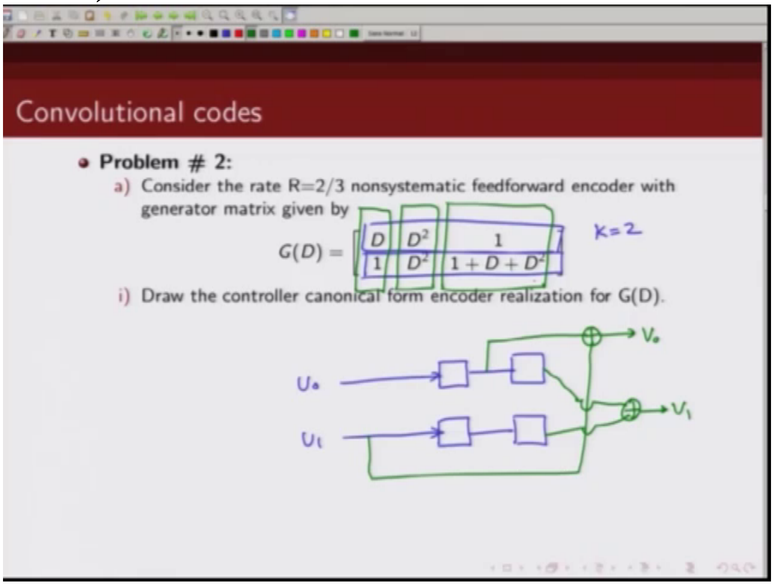
D square u 0 is this term and D square u 1 is this term, so this is your v 1

(Refer Slide Time 08:44)



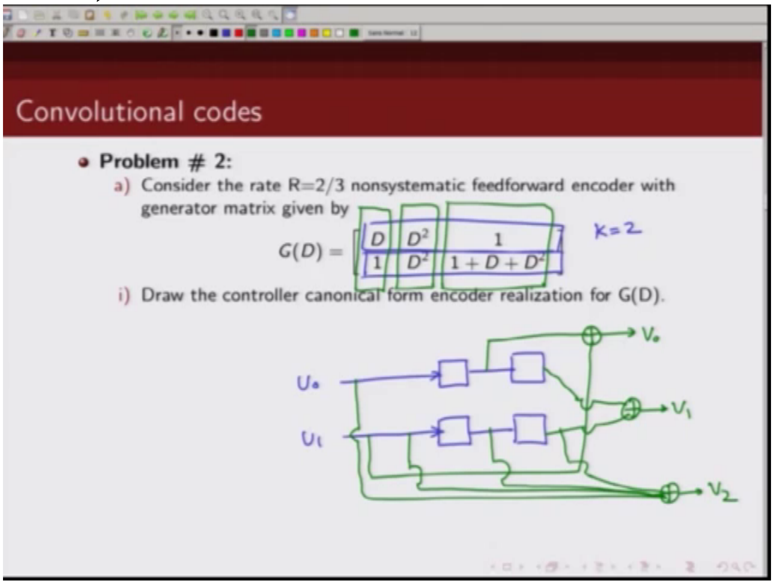
and the third output is this.

(Refer Slide Time 08:49)



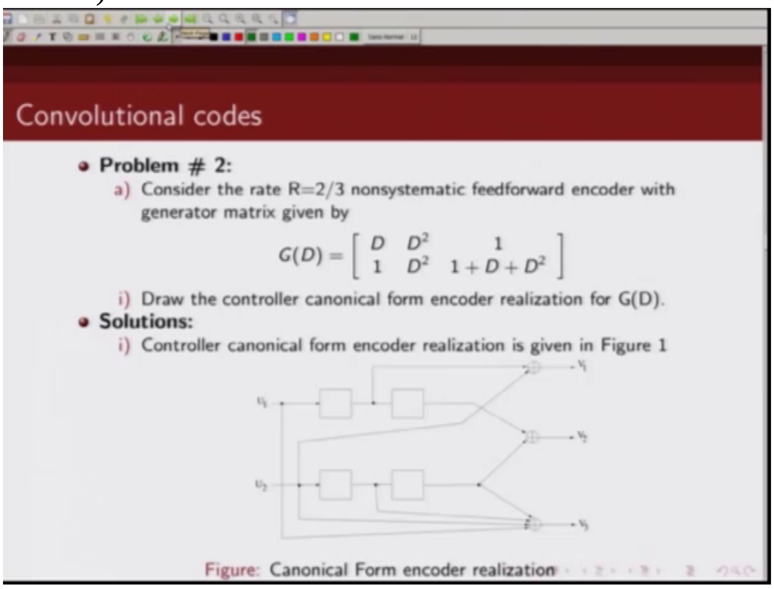
So this is, just a minute, u 1 and one D term and D square term.

(Refer Slide Time 09:12)



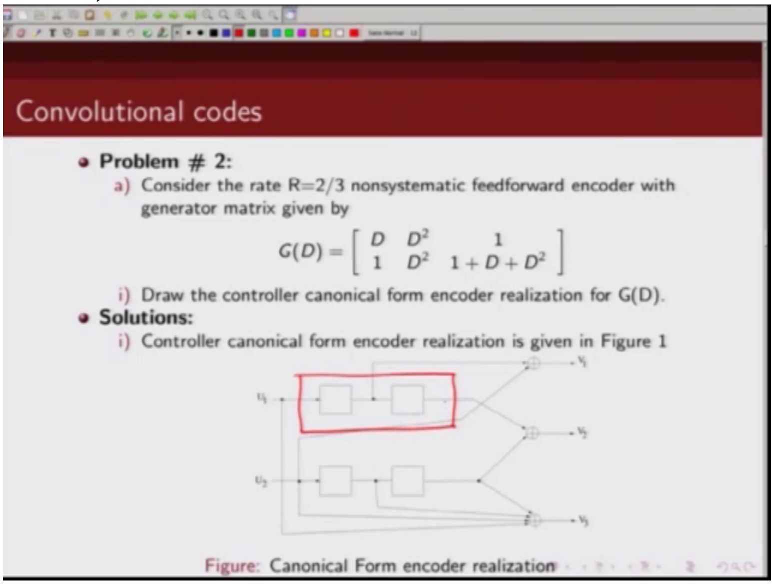
So this is your controller canonical form realization for this generator matrix. So this is

(Refer Slide Time 09:20)



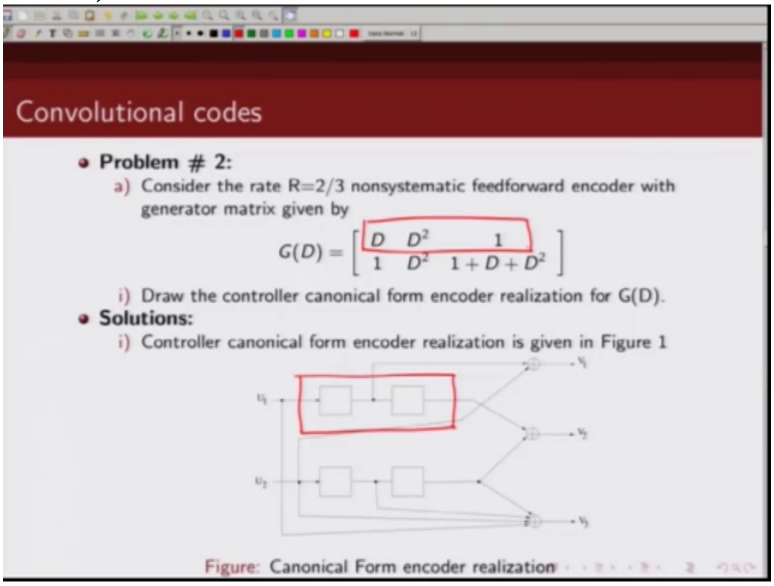
precisely what we have here. You can see, so this shift register is for

(Refer Slide Time 09:30)



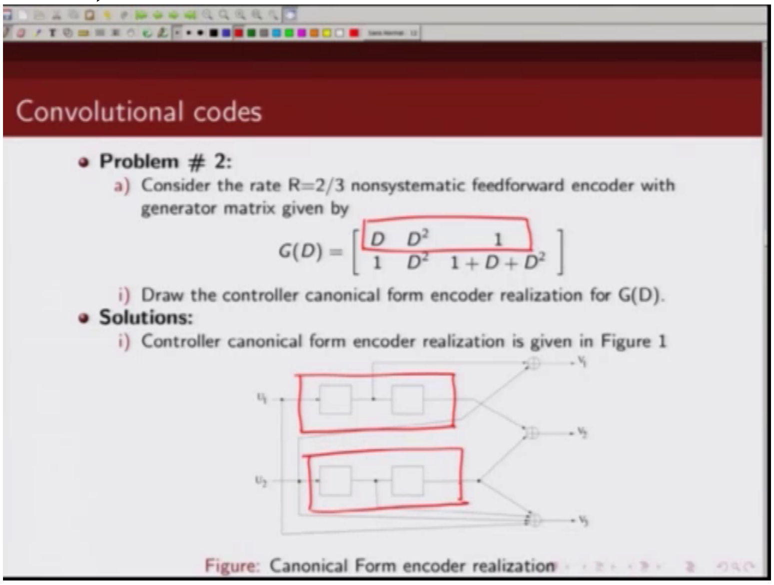
this input and this

(Refer Slide Time 09:34)



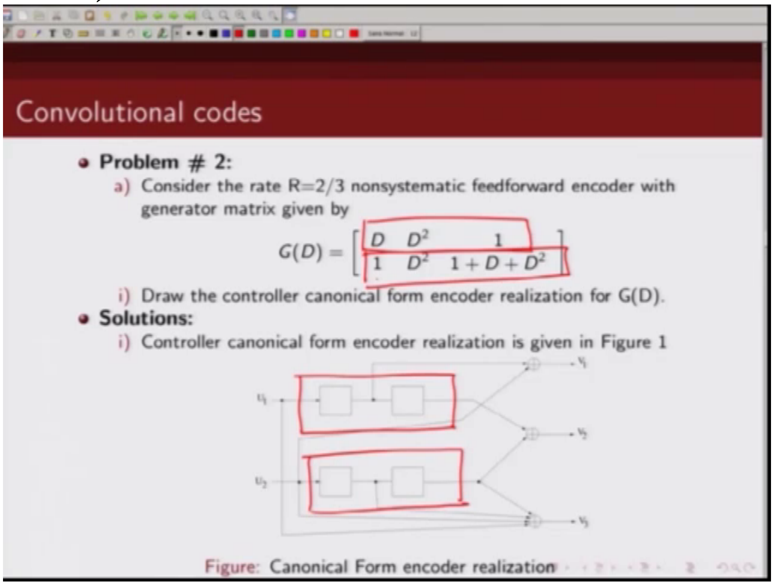
this shift register is for

(Refer Slide Time 09:37)



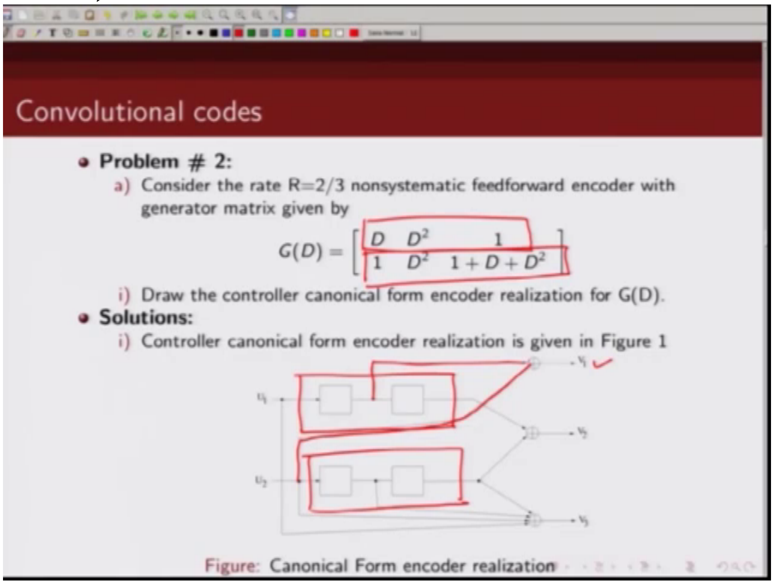
this input.

(Refer Slide Time 09:40)



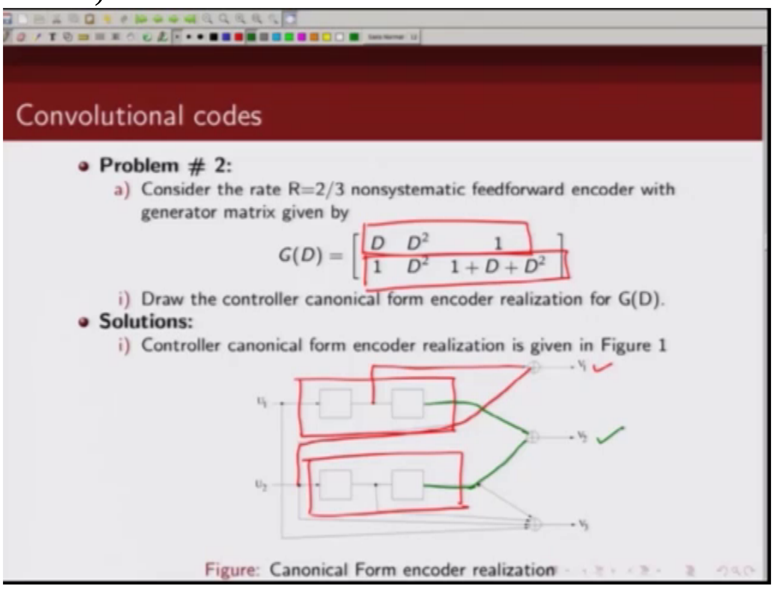
Maximum memory element for the first one is 2, second one also 2 and we can see now, the first output is D times u 1 which is this plus u 2 times this.

(Refer Slide Time 10:00)



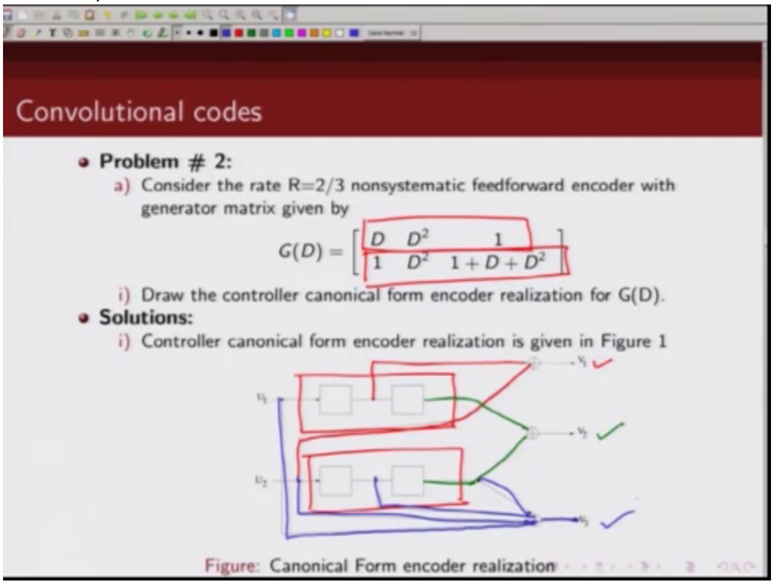
The second output is D square times u 1 and D square times u 2, so that is this.

(Refer Slide Time 10:11)



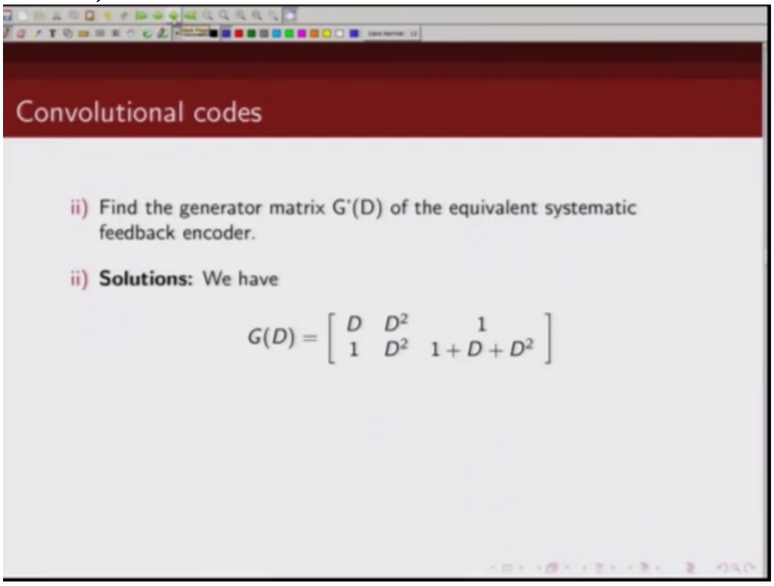
And the third output is u 1 which is this and this is u 1 D times u 1, D square times u 1, so that's your third output,

(Refer Slide Time 10:27)



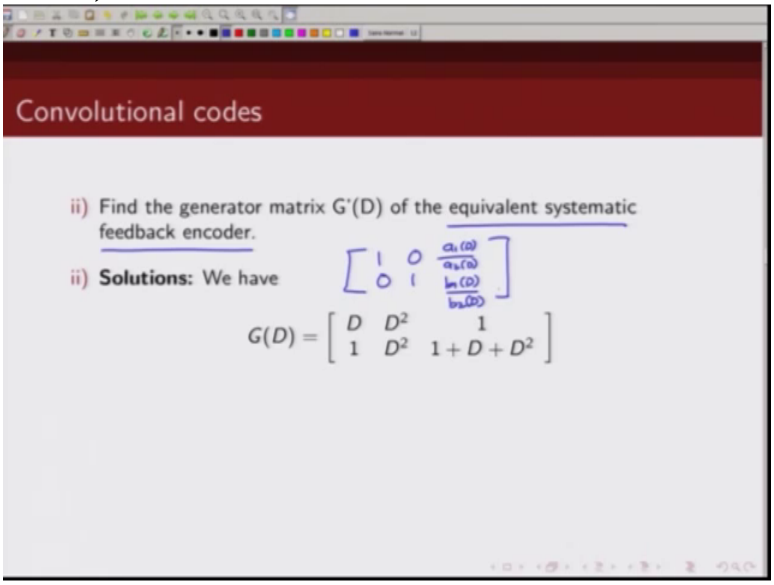
Ok.

(Refer Slide Time 10:31)



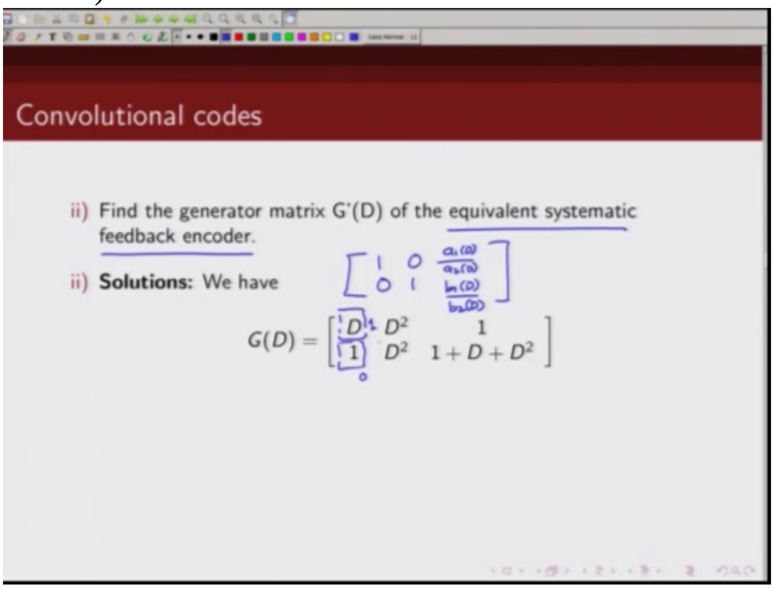
Now this was a non systematic encoder. Can we find an equivalent systematic encoder or systematic encoding matrix for this generator matrix, the answer is yes. So how do we find a systematic encoding matrix? So this has to be put in the form like this; 1 0 0 1 and some matrix here let's call it a 1 D times a 2 D and b 1 D times b 2 D. So we will have to bring

(Refer Slide Time 11:10)



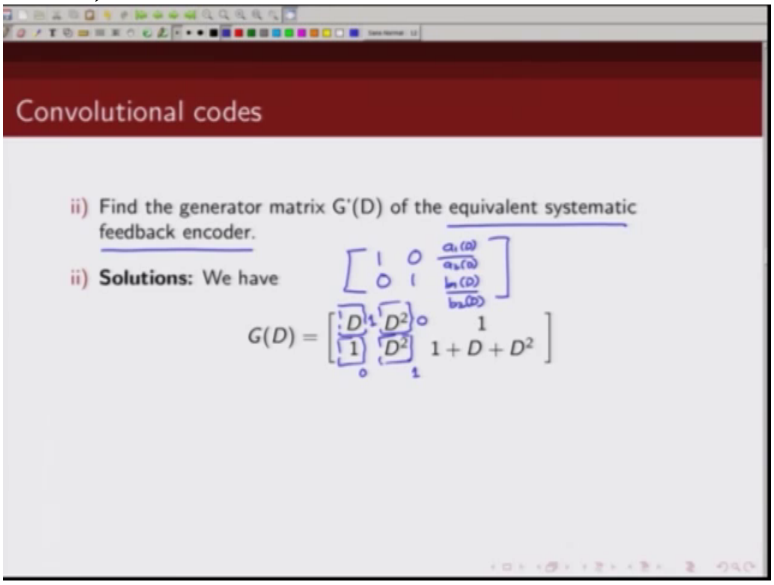
this matrix in this particular form. So we have to get this to 1, this to, this has to be changed to 1, this has to be brought to 0,

(Refer Slide Time 11:22)



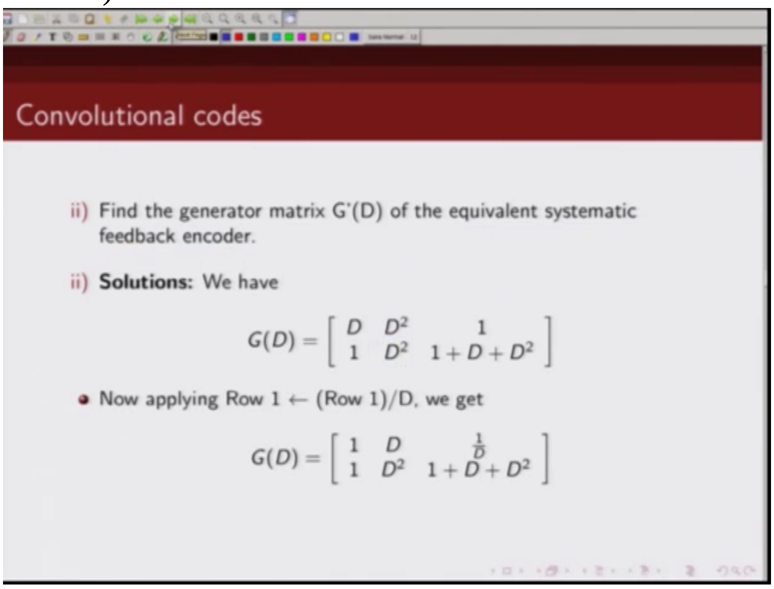
, this has, this we have to brought to 1 and this we have to bring to

(Refer Slide Time 11:28)



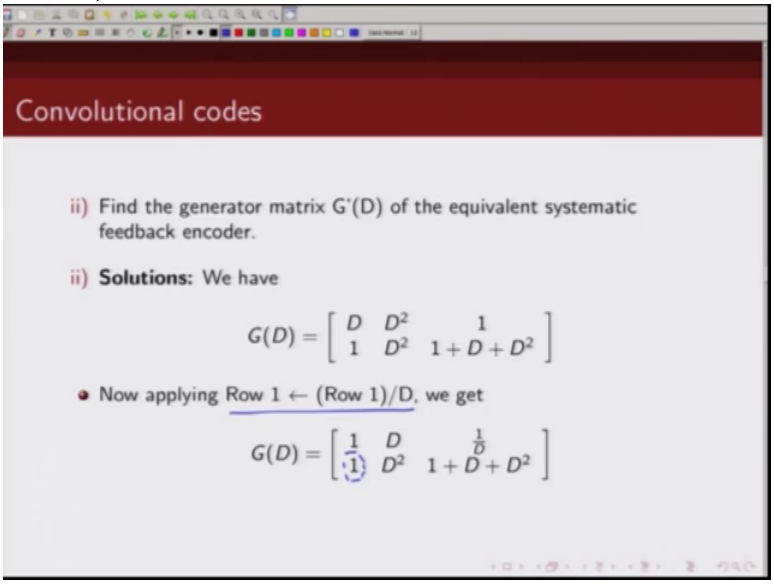
0. Now we will do elementary row operation to get an identity matrix here. So let's do that.

(Refer Slide Time 11:37)



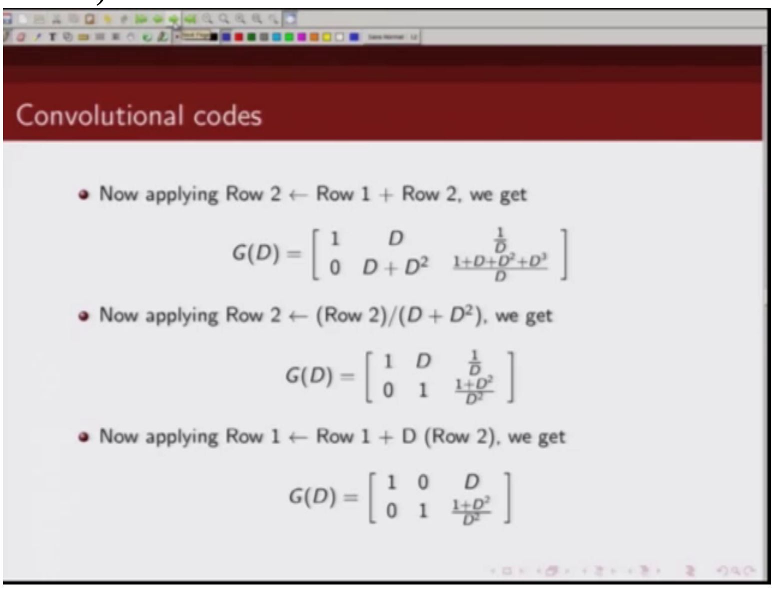
So first thing that we do is, we make this a 1. How do we make this a 1? We do this transformation that row 1 is, row 1 by D. So we divide this whole thing by D, what we get is 1 D 1 by D. Next we would like to get a zero here.

(Refer Slide Time 12:04)



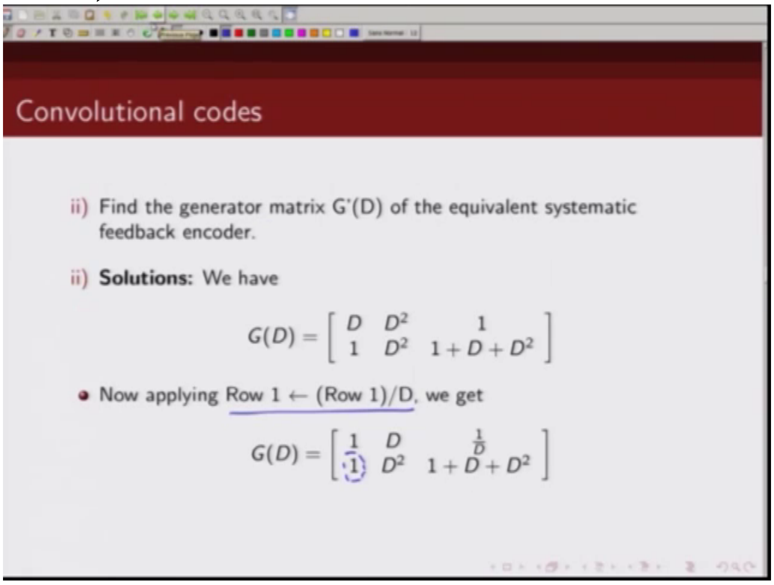
Here we would like to get a zero. How can we get a zero here?

(Refer Slide Time 12:09)



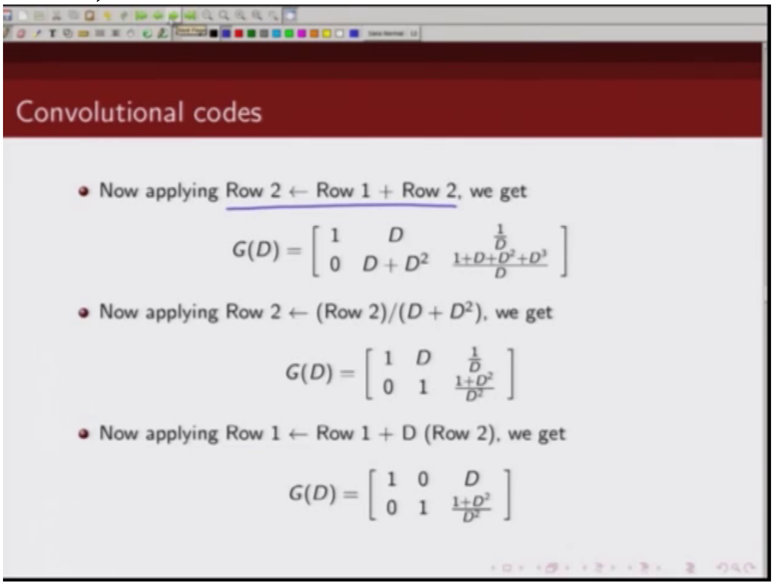
So we will do this transformation, row 2 is row 1 plus row 2. If we do that, so we

(Refer Slide Time 12:21)



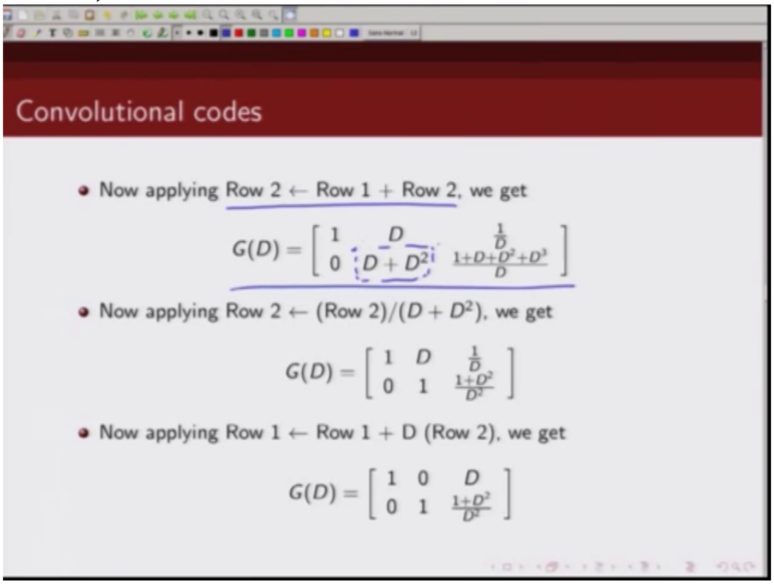
add these two, this will become 0, this will become D plus D square and what we will get is

(Refer Slide Time 12:28)



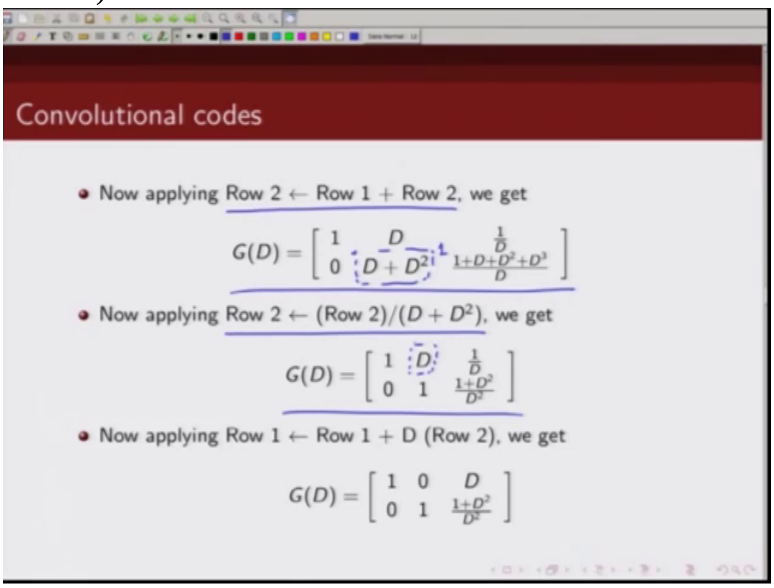
this. Next we would like to get a 1 here,

(Refer Slide Time 12:36)



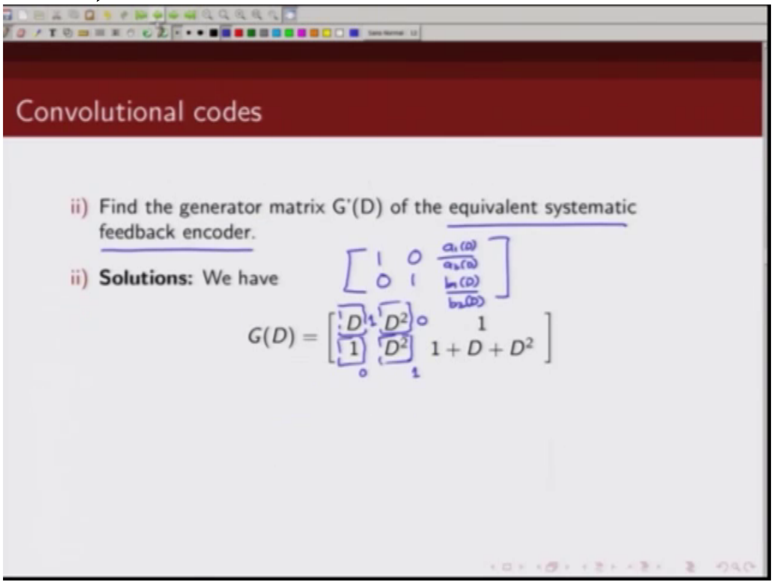
like to get a 1 here. How can we get a 1 here? We divide row 2 by this. So we do this transformation that row 2 is row 2 divided by D plus D square and once we do that, we get this. Next we would like to get a zero here.

(Refer Slide Time 12:58)



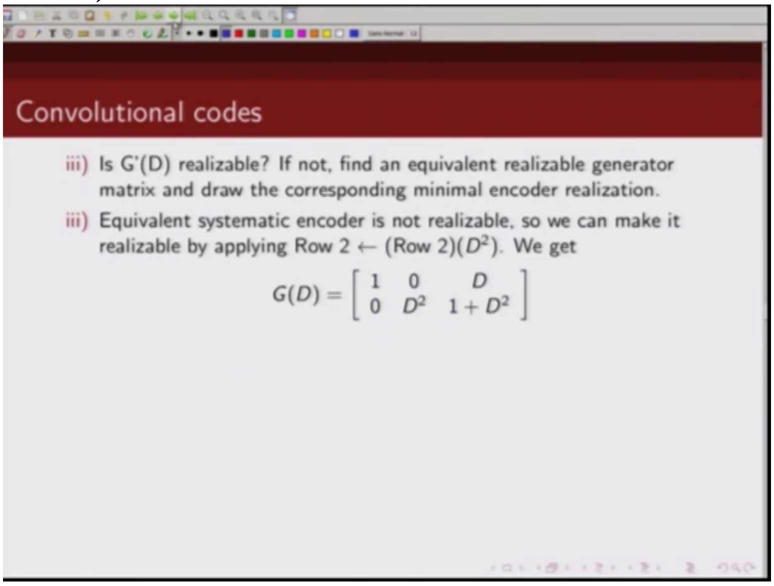
How do we get a zero here? We multiply row 2 by D and add it to row 1. So we do this transformation that row 1 is row 1 plus D times row 2. And when we do that, we get this. So this is our equivalent systematic encoder for the

(Refer Slide Time 13:27)



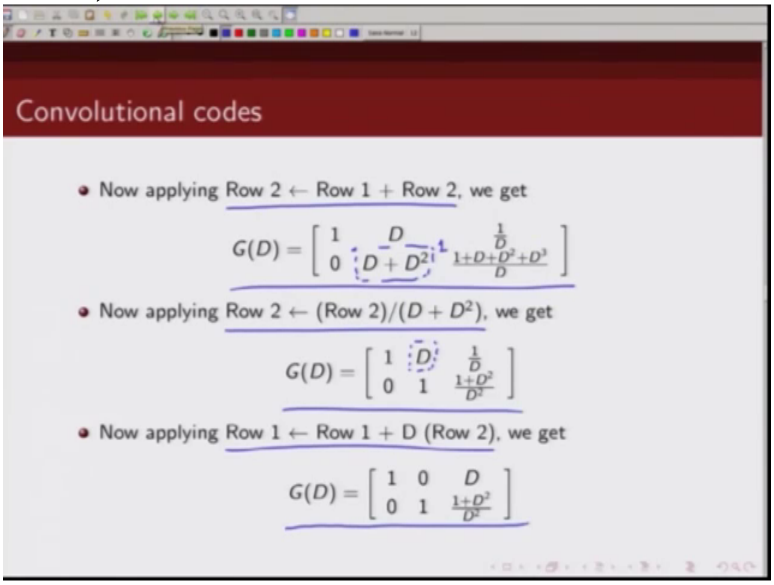
generator matrix this, Ok.

(Refer Slide Time 13:33)



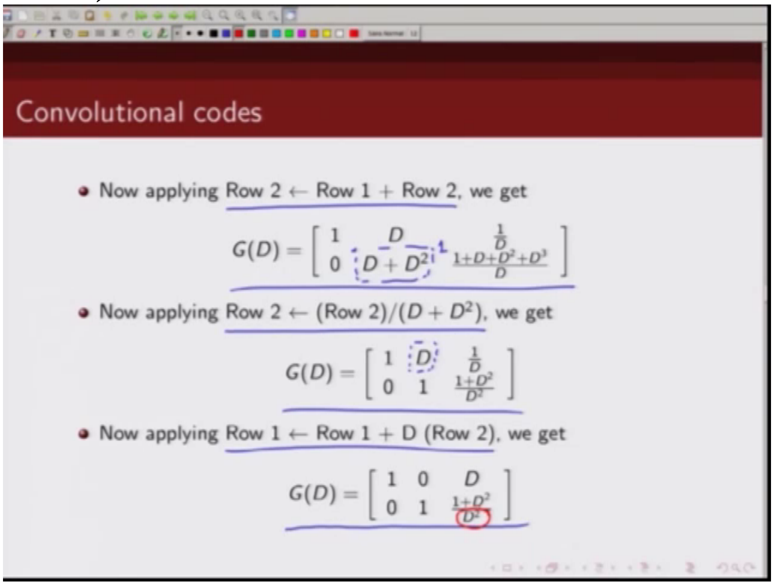
Next. Is this equivalent systematic generator matrix, is it realizable? If it is not, find out an equivalent realizable generator matrix and draw its corresponding minimal encoder realization. Now note here

(Refer Slide Time 13:52)



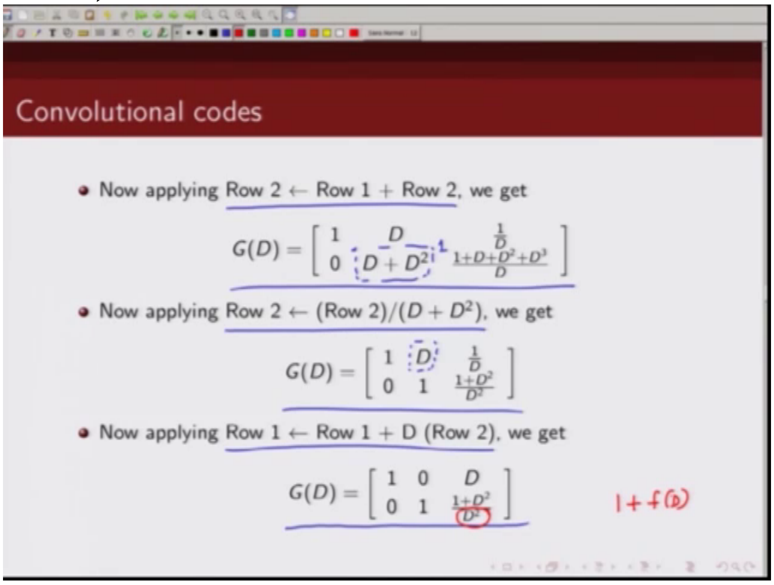
this generator matrix has a term 1 plus

(Refer Slide Time 13:58)



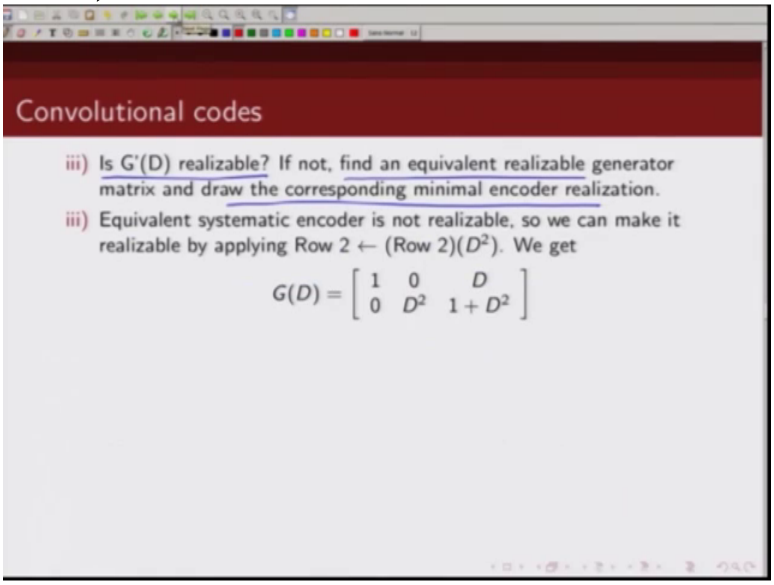
D square in the denominator. Now this cannot be realizable. So any denominator term that we have, it has to be of the form 1 plus some polynomial here

(Refer Slide Time 14:13)



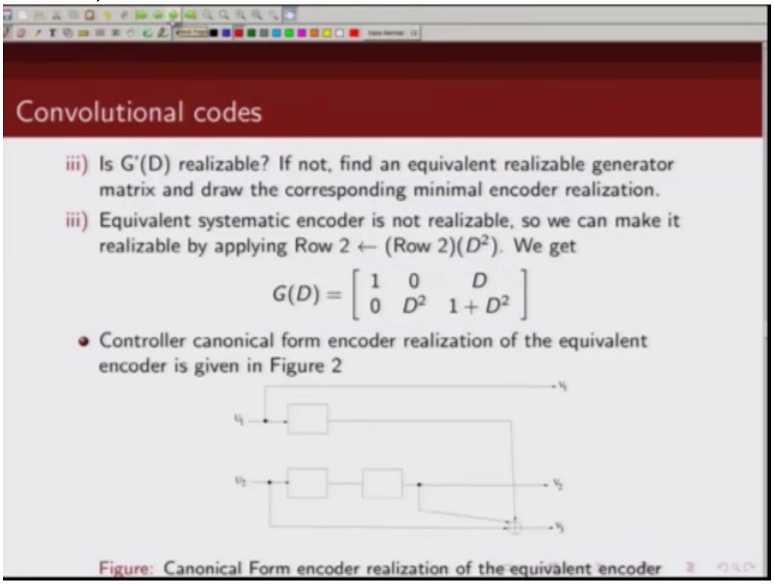
but here this 1 is not here. So we cannot realize a rational function of this form using our shift register. So this particular equivalent systematic encoder is not realizable. However if we multiply this by D square, what we

(Refer Slide Time 14:33)



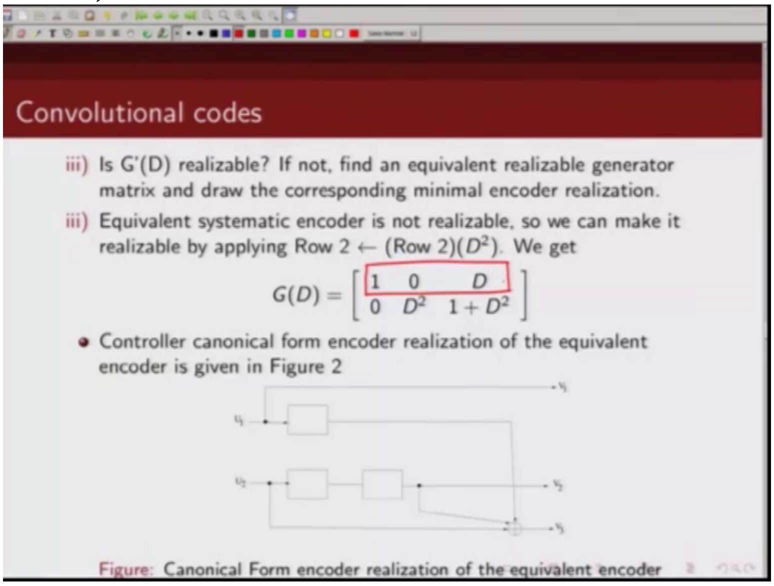
get if we do this transformation, what we get is this. This is no longer, so what we are getting now is basically a new equivalent encoder which is in the feed forward form and it is realizable.

(Refer Slide Time 14:54)



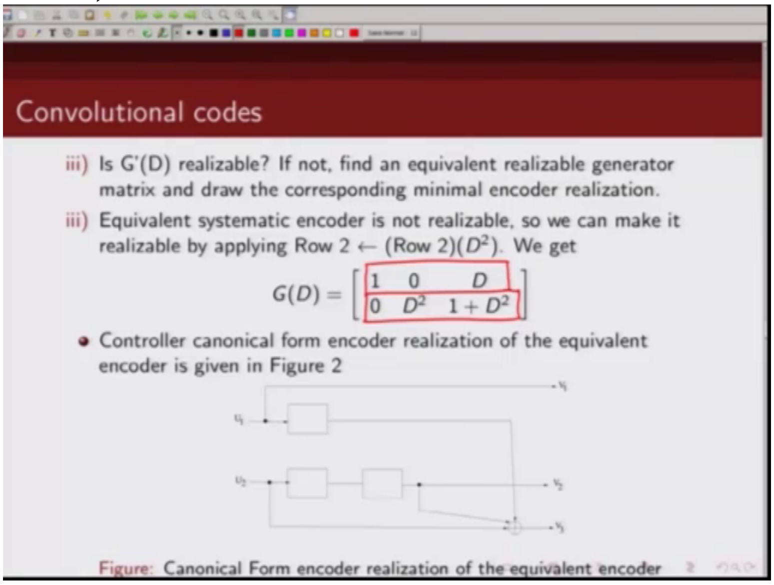
So how do we realize it? Again if we using controller canonical form realization we will have one set of shift registers for this

(Refer Slide Time 15:02)



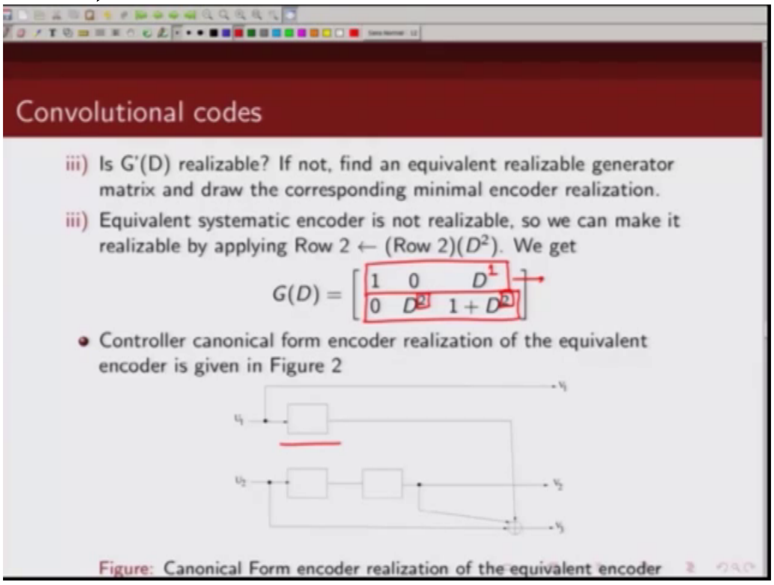
input, another set of shift registers for this input.

(Refer Slide Time 15:06)



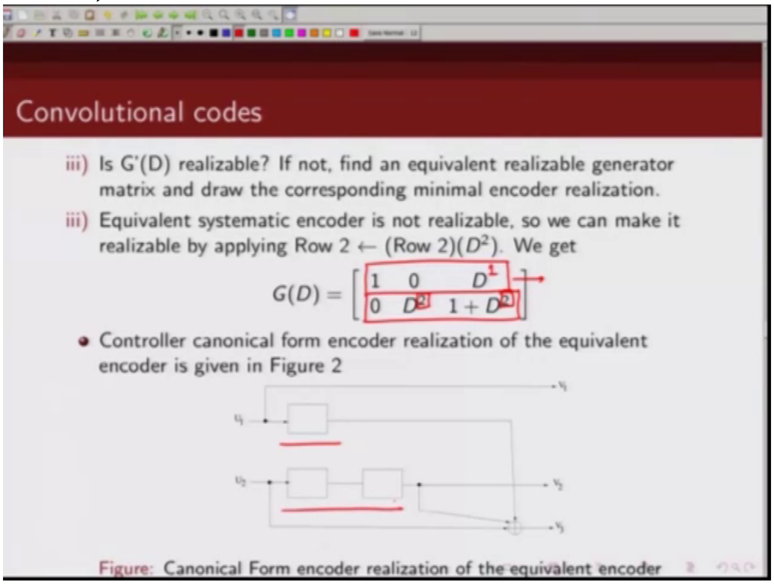
What's the maximum memory for the first row? The maximum power of D is 1. So we will have 1 memory element for the first input and what's the maximum power of D for the second? That's 2 here, 2 here, 2 here

(Refer Slide Time 15:22)



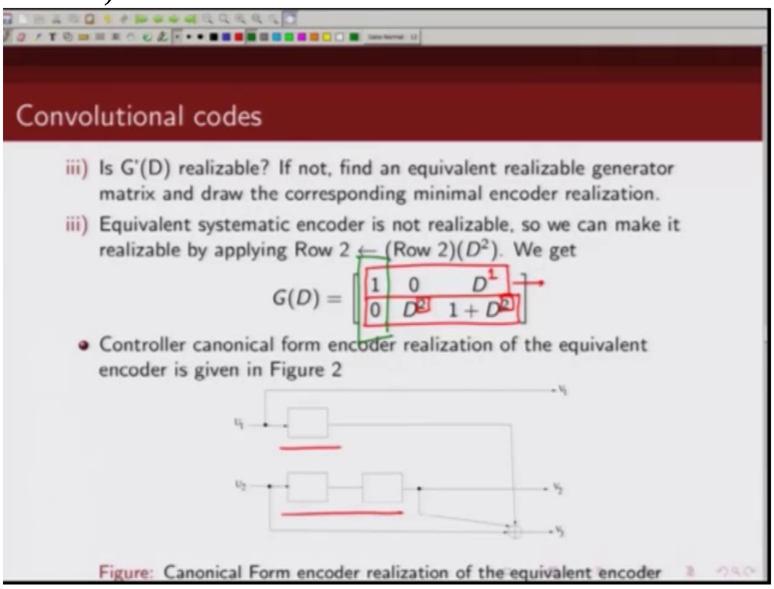
so we will use 2 memory element

(Refer Slide Time 15:25)



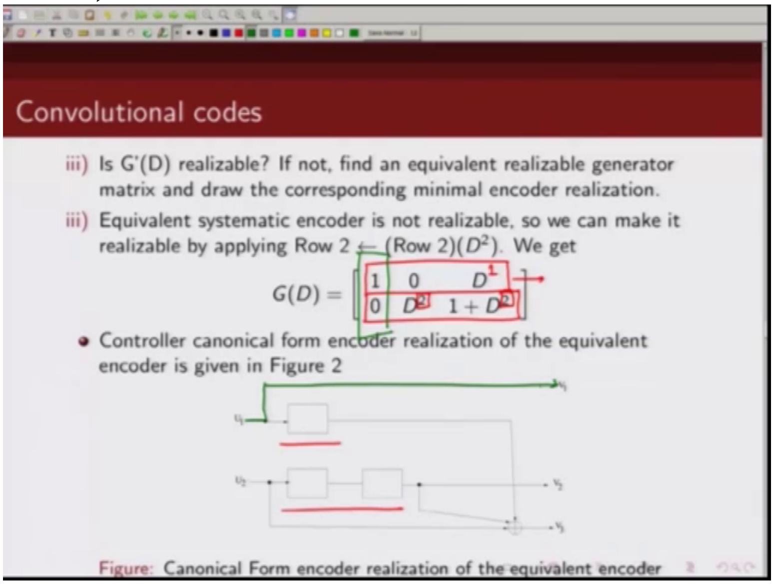
for the second input. And again what are our outputs? There are 3 outputs. The first output is this,

(Refer Slide Time 15:34)



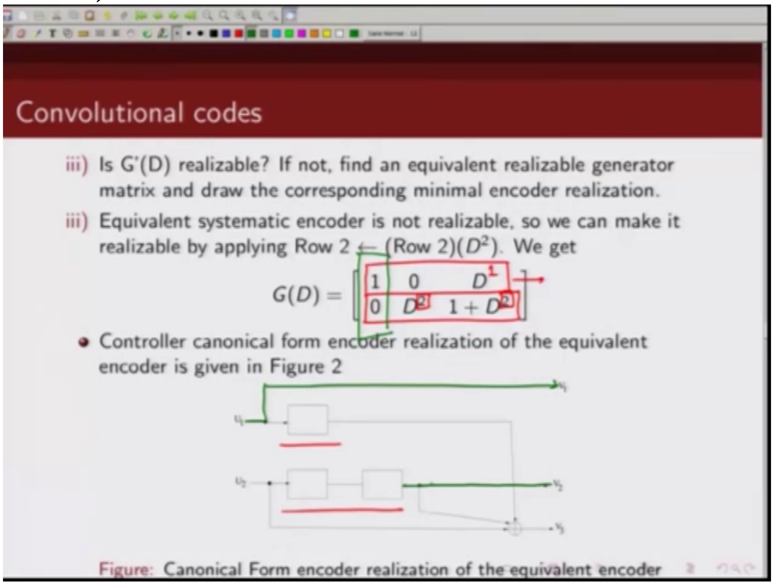
this is u 1 times, this is just u 1 so this is this, second one is D square

(Refer Slide Time 15:42)



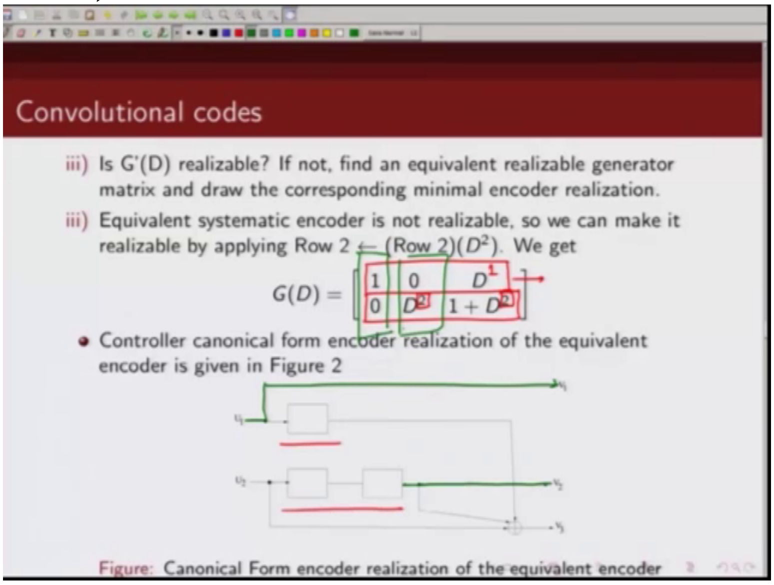
of u 2. So D square of u 2 is just this term.

(Refer Slide Time 15:49)



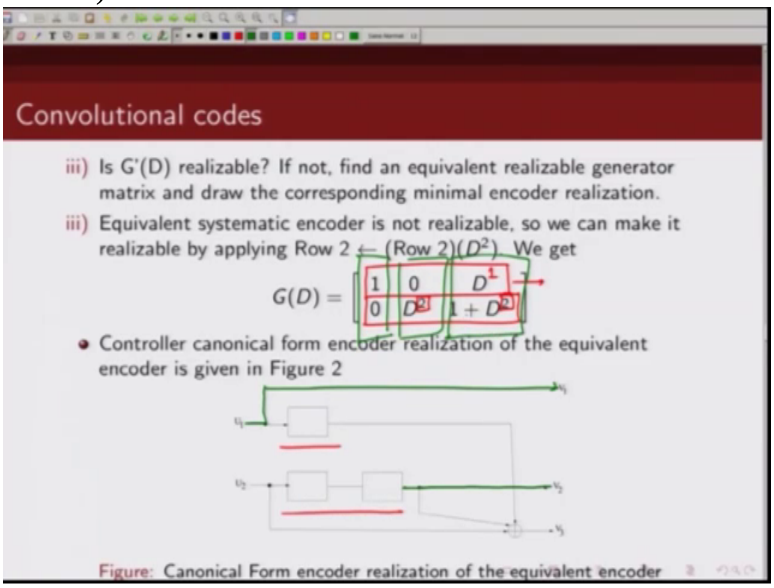
So this is my second output and the third

(Refer Slide Time 15:53)



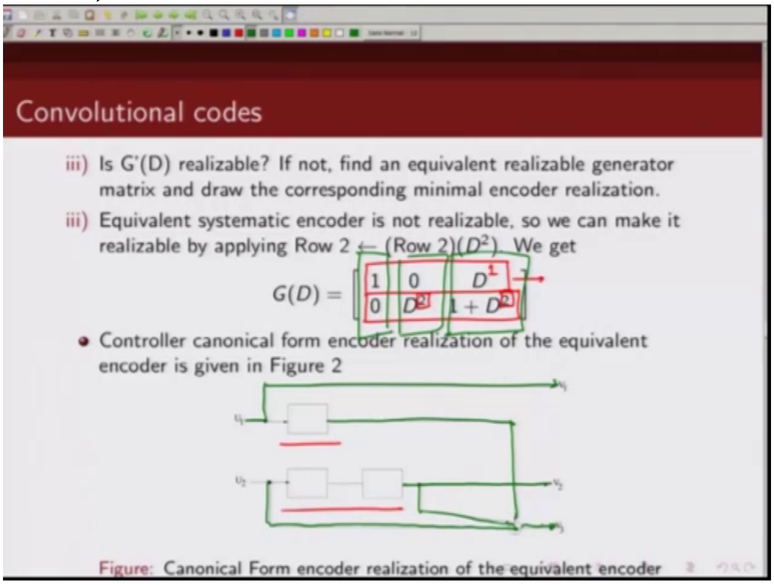
output is this,

(Refer Slide Time 15:56)



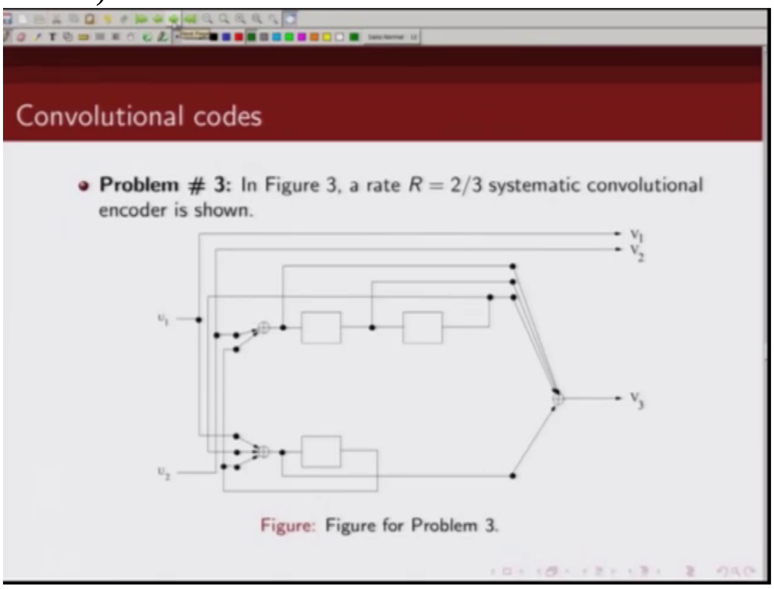
D times u 1 D which is this one and one times u 2 D and D square times u 2 D, so that's this. This is our third

(Refer Slide Time 16:12)



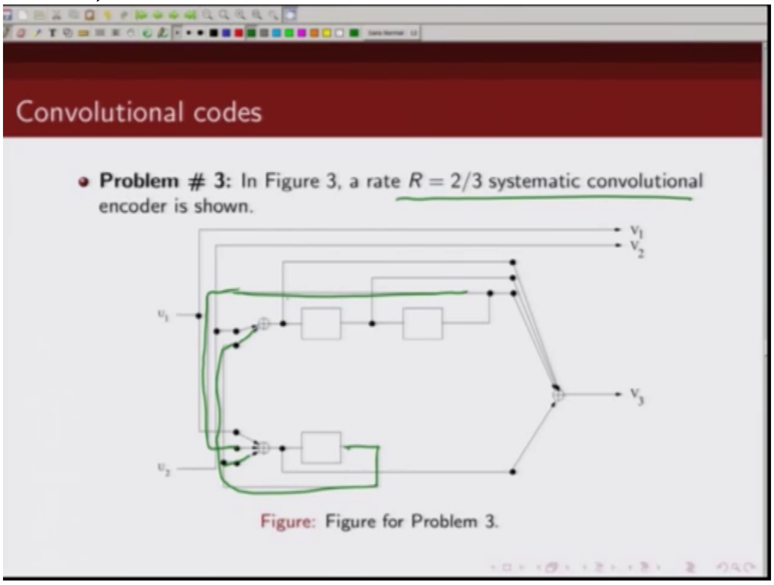
output. Now given

(Refer Slide Time 16:15)



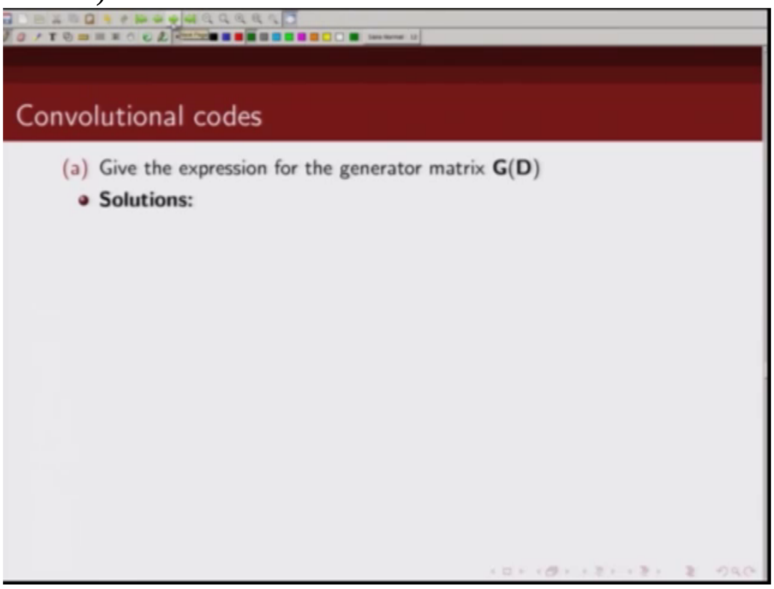
below is a rate two third systematic convolutional encoder. Please note this is neither in the controller canonical form realization or in the observer canonical form realization. Note here the feedback terms that are coming here are not only coming from the same encoders like this, feedback is not only, so if you look at the feedback, feedback from this is going to this encoder and feedback from here is going to this encoder. So not only feedback is coming to the same encoder but it is also going to the other encoder.

(Refer Slide Time 16:55)



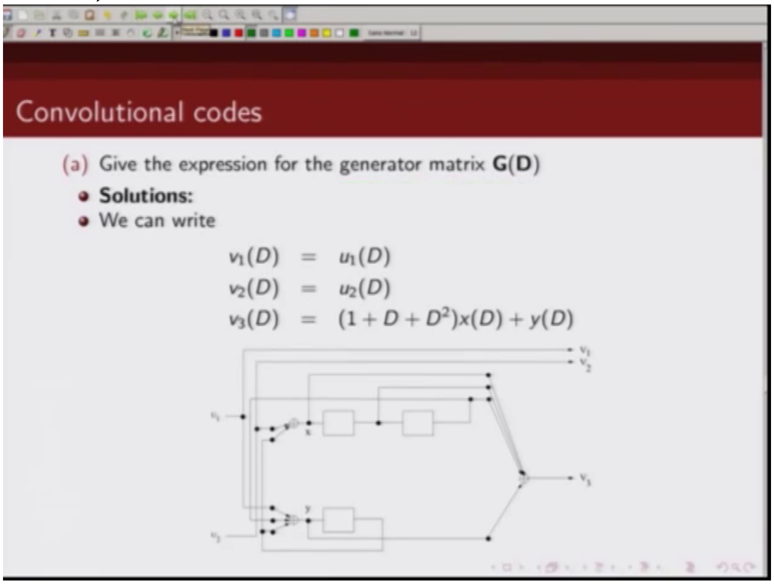
So this realization is

(Refer Slide Time 16:58)



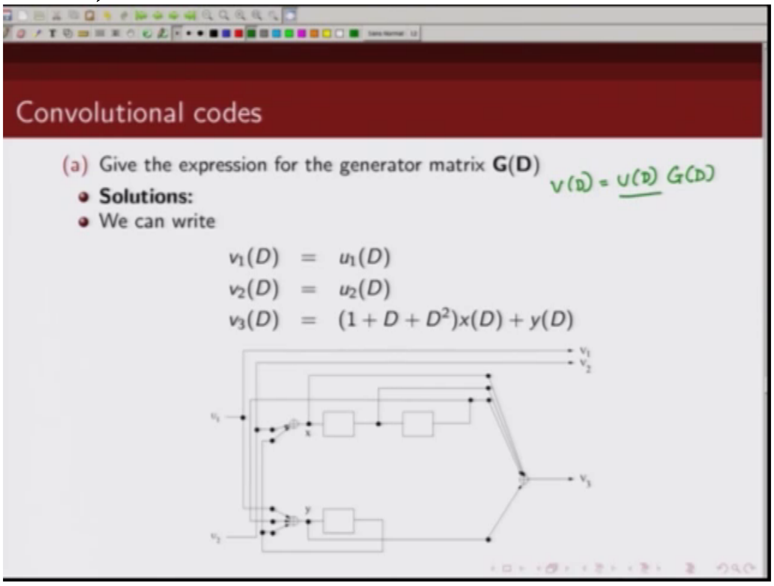
a very compact realization. The question that has been asked is can you find out the generator matrix corresponding to this encoder? So

(Refer Slide Time 17:10)



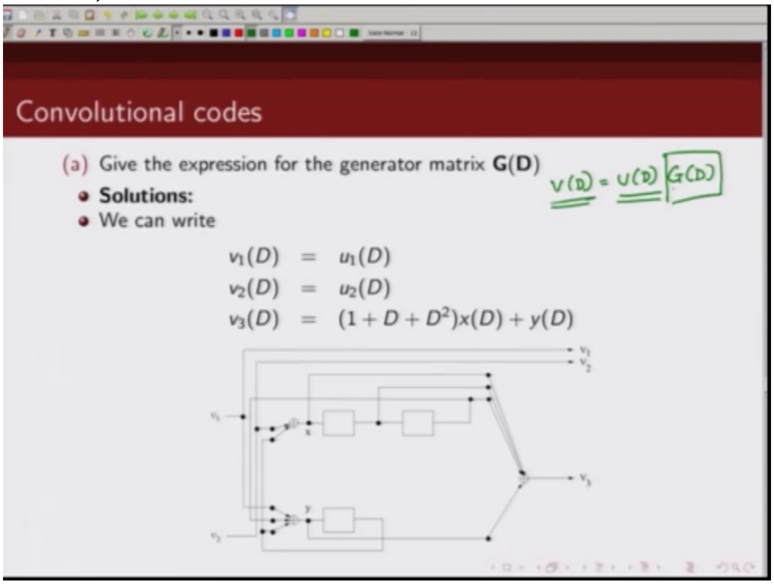
how do we find the generator matrix? We know this is a relation between the input and the output. So how these

(Refer Slide Time 17:20)



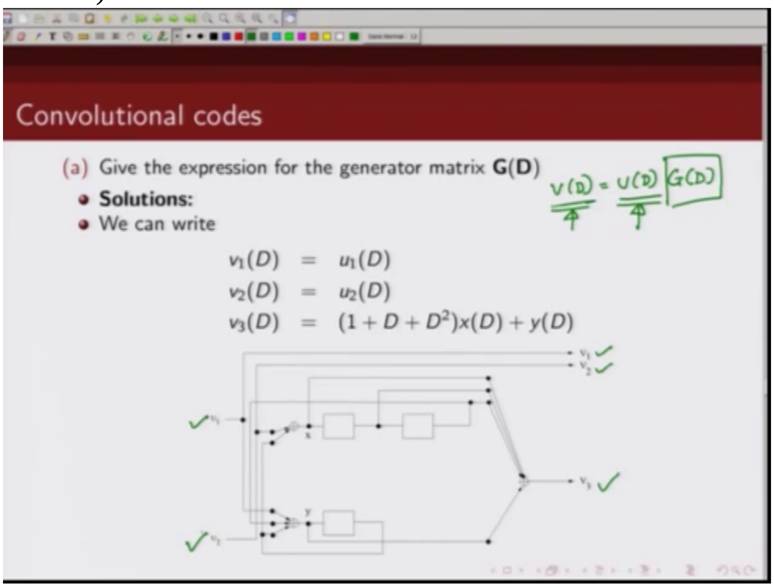
inputs are getting mapped to the output, that is governed by this generator matrix.

(Refer Slide Time 17:26)



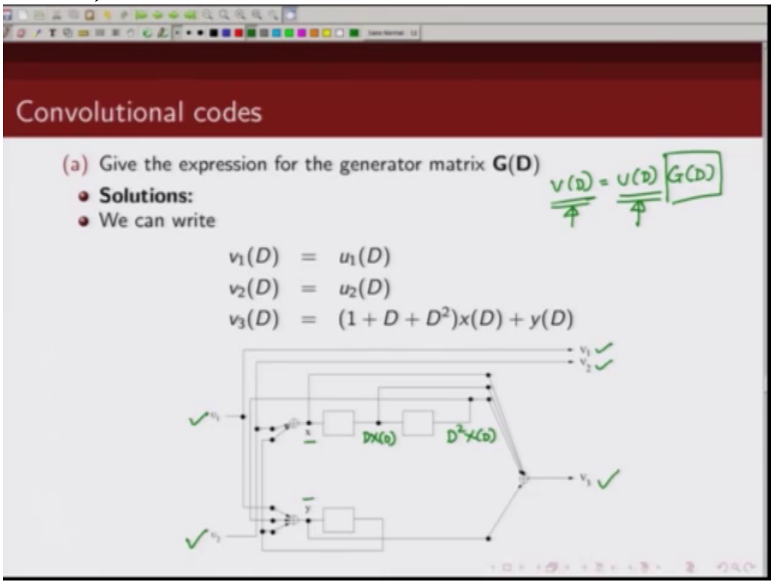
So what we are going to do is we are going to write the output v D in terms of input u D. And then that would give us our generator matrix. So our objective is to write v 1, v 2, v 3 in terms of u 1 and u 2,

(Refer Slide Time 17:50)



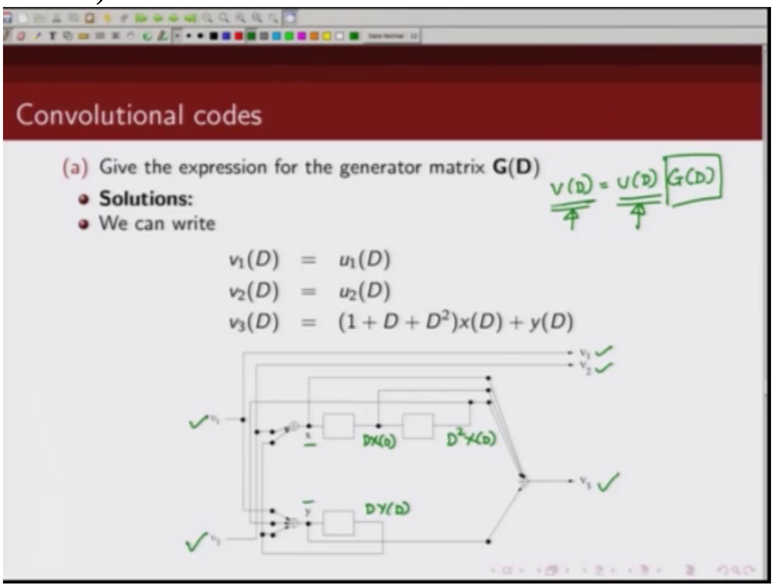
fine. We use some auxiliary variables; x and y which basically will help us find the contents here. So if this is x of D, this term will be D times x of D and this will be D square times x times D.

(Refer Slide Time 18:11)



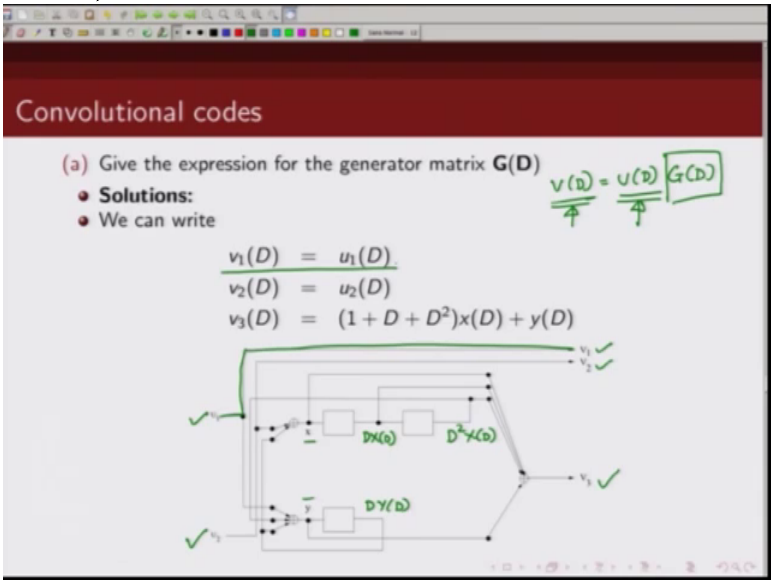
Similarly if this is y, this term will be D times y of D.

(Refer Slide Time 18:16)



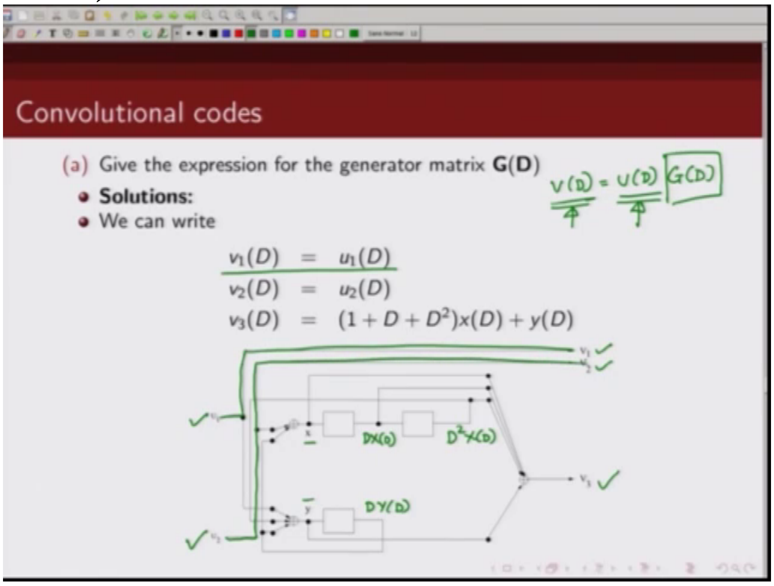
So what is v 1 of D? v 1 of D is u 1 of D, you can see u directly goes, this input directly goes here. So v 1

(Refer Slide Time 18:30)



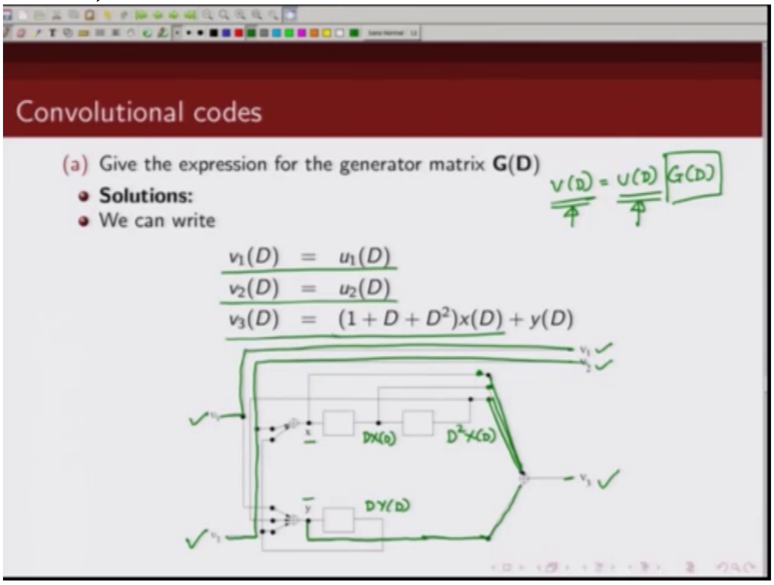
of D is u 1 of D. Similarly this input u D directly goes to the output here.

(Refer Slide Time 18:43)



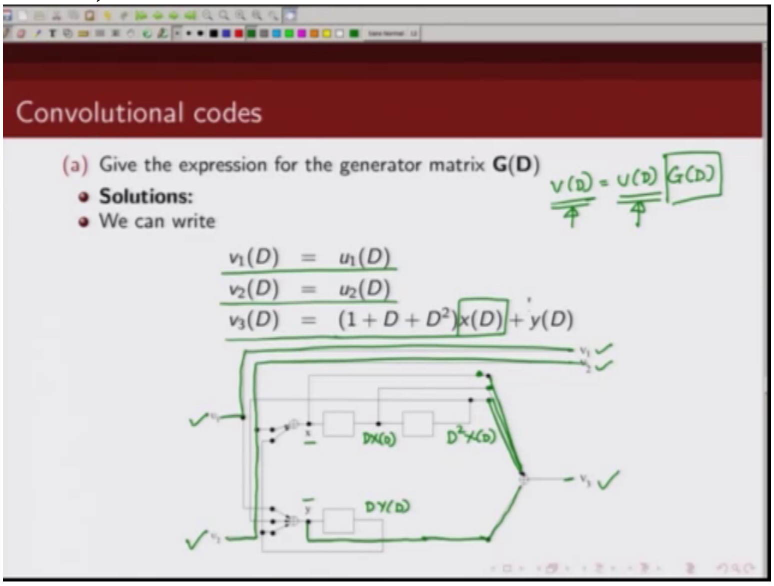
So v 2 of D is u 2 of D. Now what is v 3 of D? v 3 of D is this term which is x of D, this term D times x of D and this term which is D square x of D. So this is this term plus this term. So it is these 3 terms. Now what is this term? This is y of D. So we have

(Refer Slide Time 19:17)



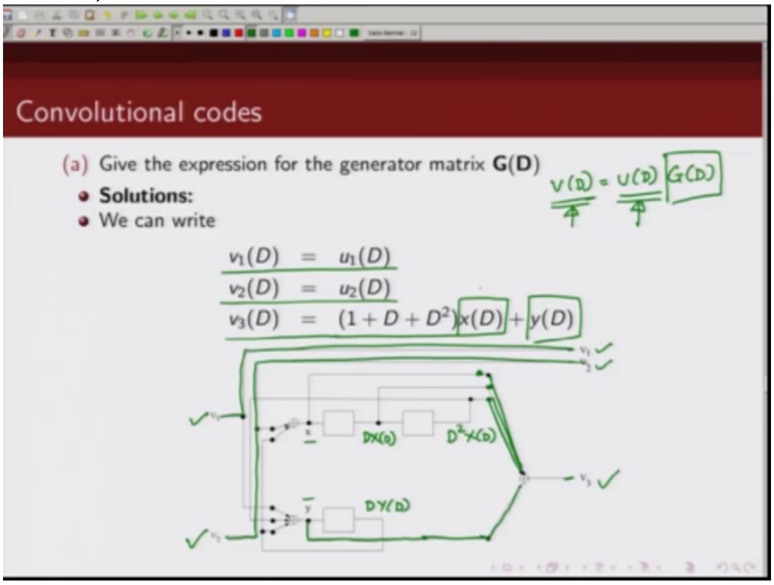
written v 1 of D, v 2 of D, v 3 of D in terms of u 1, u 2, x of D and y of D. Now note we need to get rid of x of D and

(Refer Slide Time 19:30)



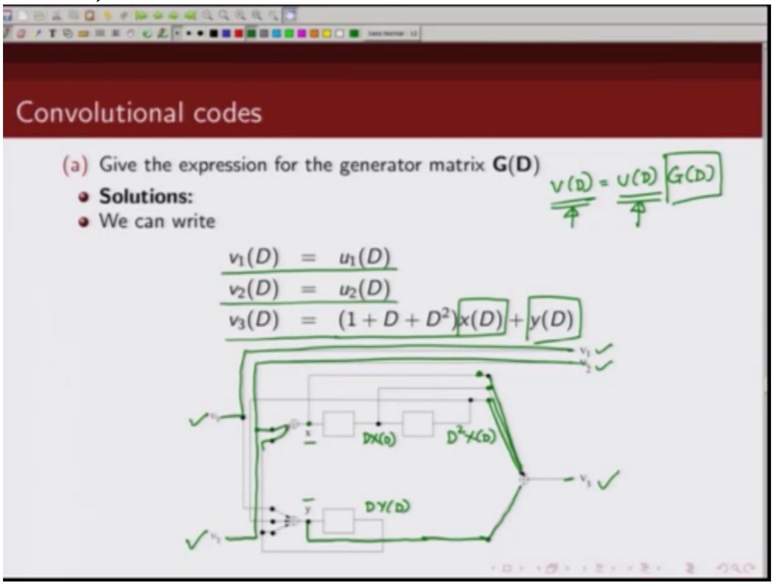
y of D, and we have to

(Refer Slide Time 19:32)



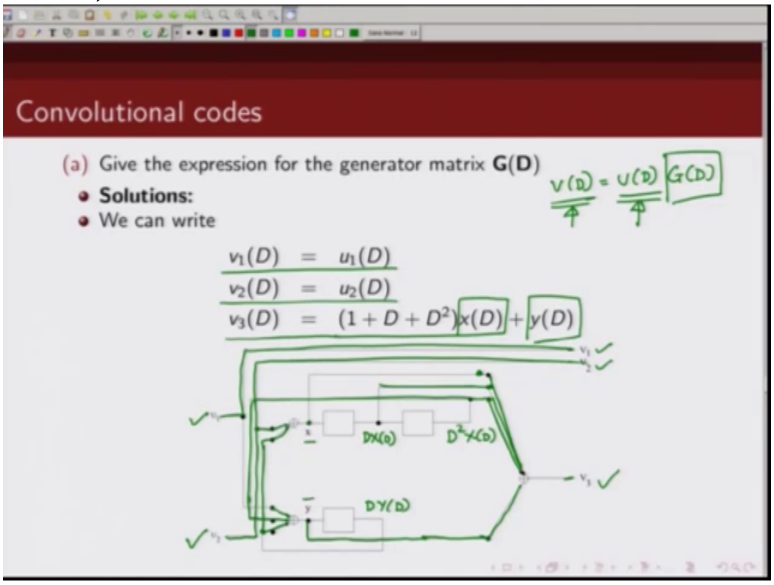
write these in terms of u 1 and u 2. Now what is x of D? x of D is this and this. Similarly what is y of D?

(Refer Slide Time 19:44)



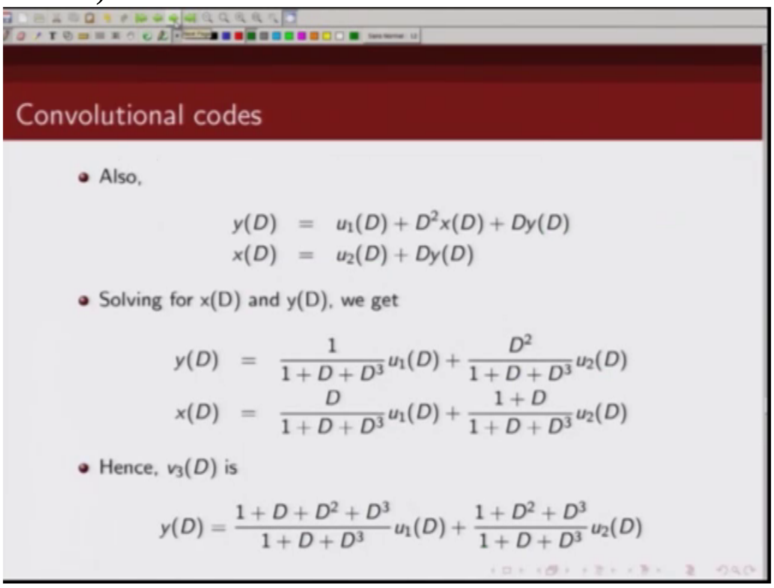
y of D is this term, this, this term sorry this term and this term, Ok. So we can write

(Refer Slide Time 20:01)



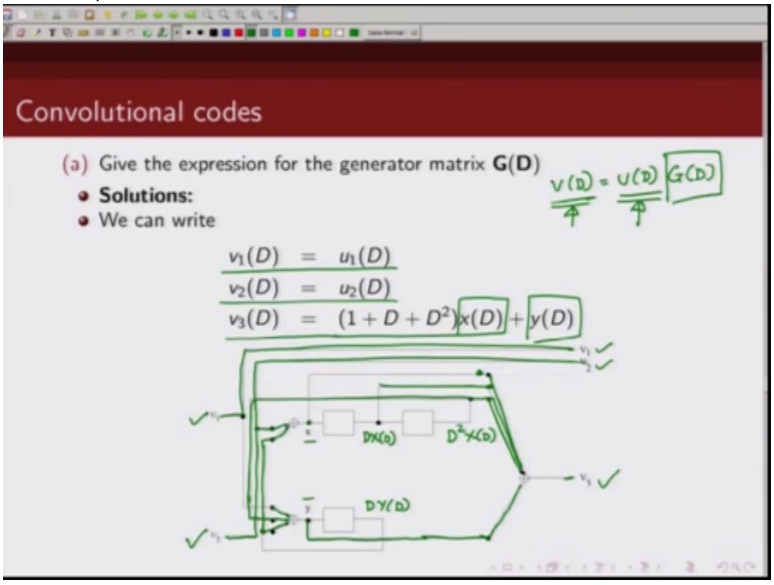
2 more equations

(Refer Slide Time 20:02)



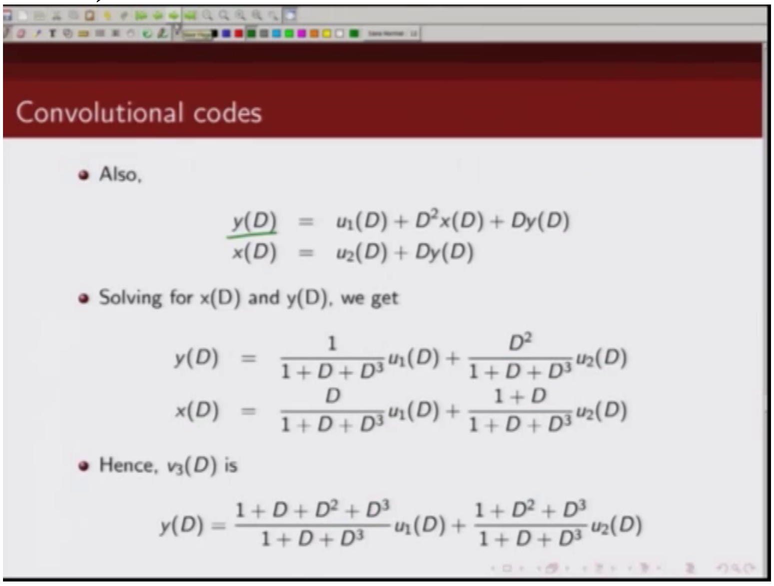
for x of D and y of D. So again y of D as I said is u 1 of D,

(Refer Slide Time 20:10)



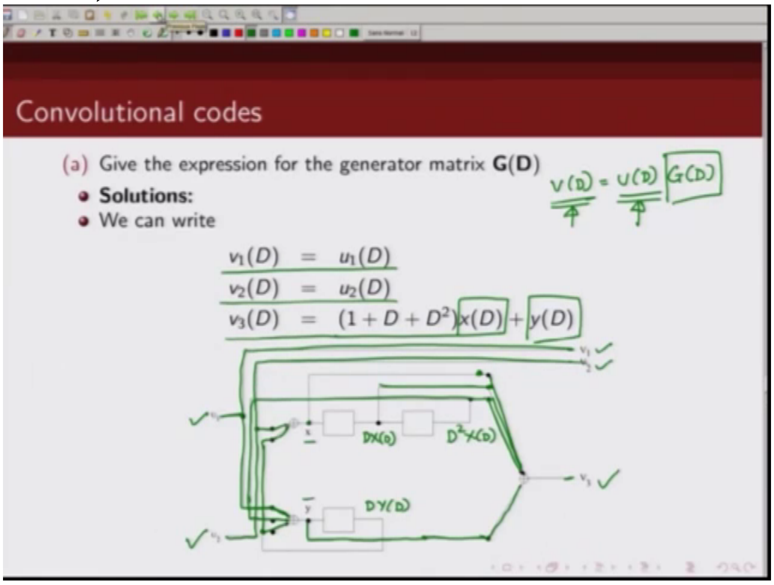
y of D is u 1 of D which is this one, this is u 1 of D

(Refer Slide Time 20:21)



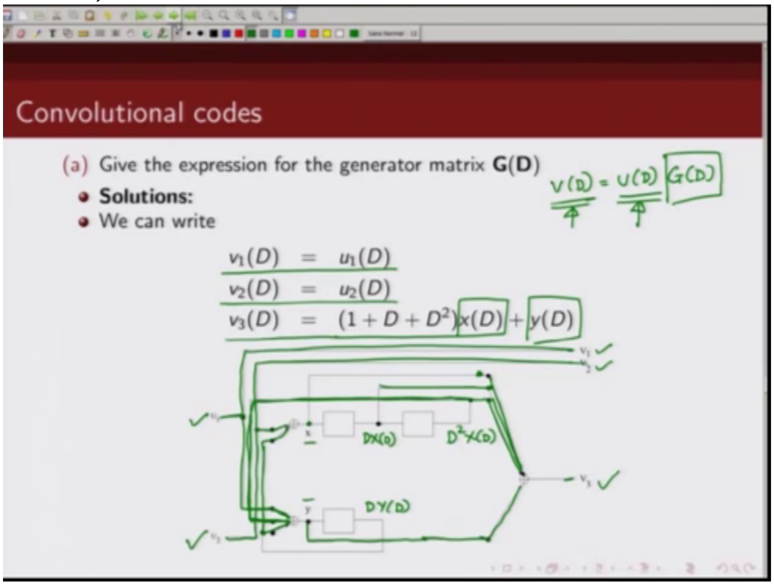
plus D square x of D.

(Refer Slide Time 20:24)



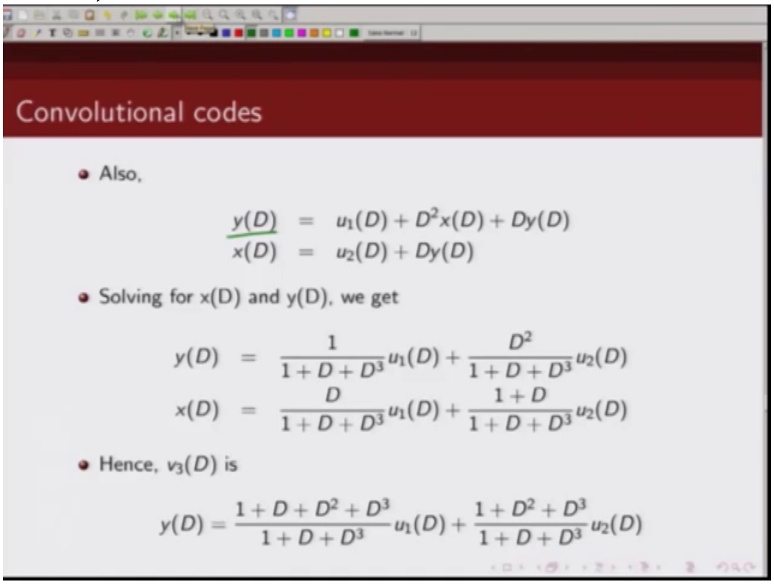
D square x of D is this term, D square of x of D is this term which is coming here, this term and there is another term

(Refer Slide Time 20:36)



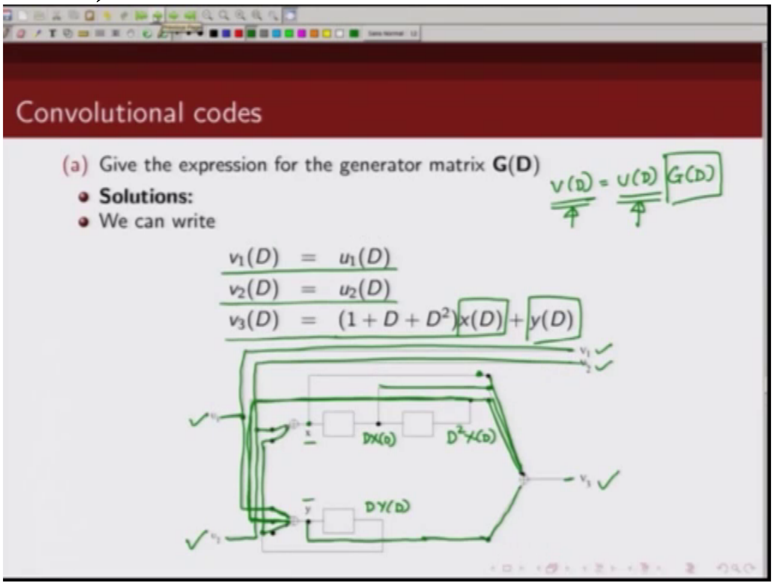
here

(Refer Slide Time 20:38)



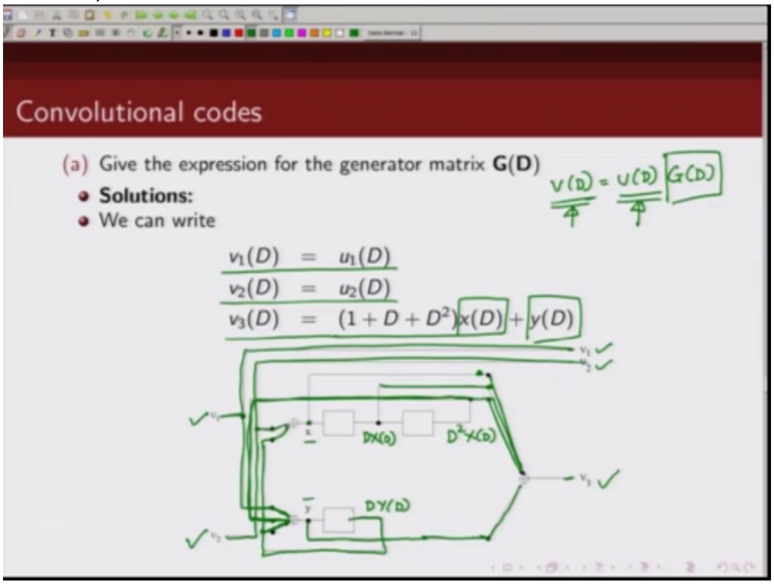
which is D times y of D. So D

(Refer Slide Time 20:41)



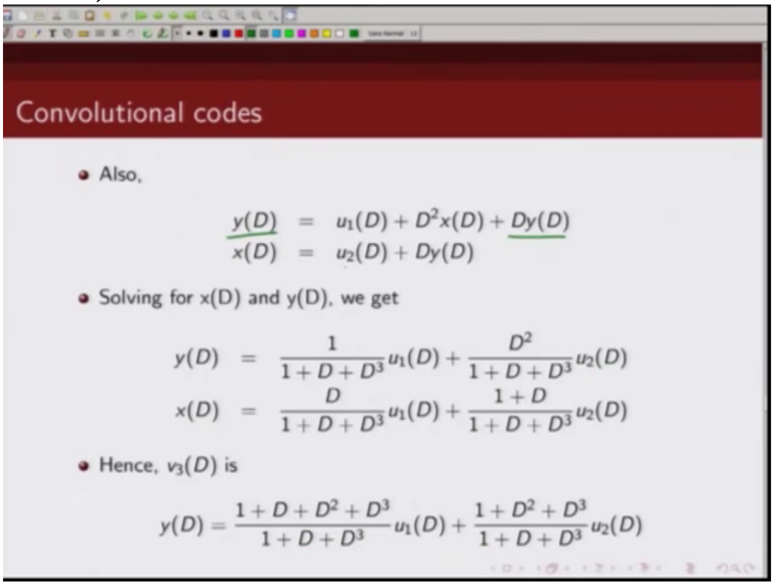
times y of D, note here, the third input here is this one which is D times y

(Refer Slide Time 20:47)



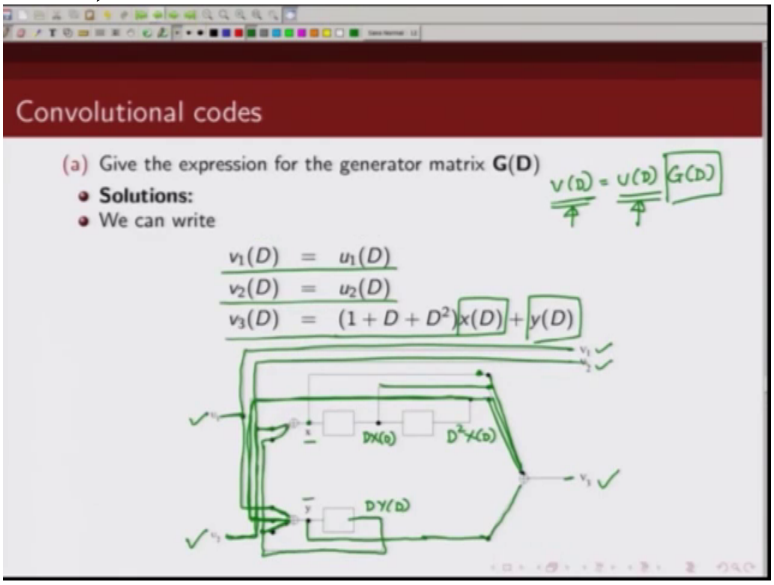
of D. Similarly x of D is, first one is this term which is u of D, so this is u

(Refer Slide Time 20:58)



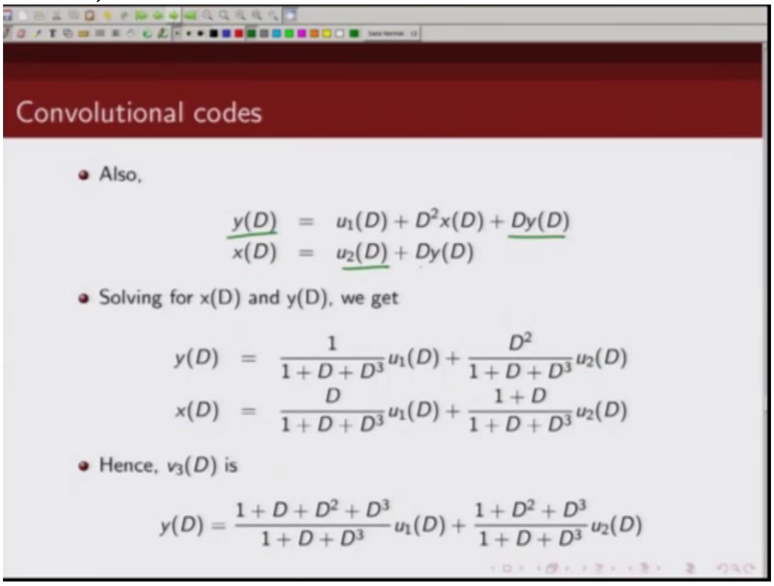
of D and the second term is

(Refer Slide Time 21:01)



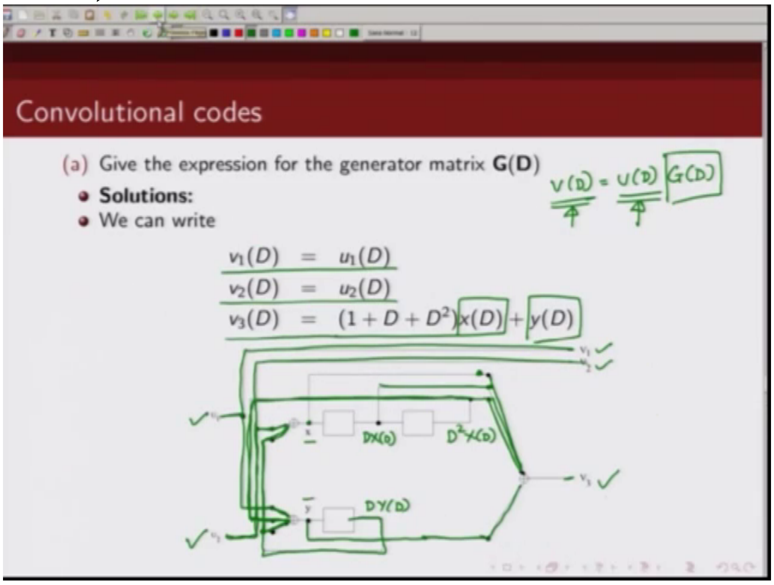
this term which is D times y of D,

(Refer Slide Time 21:06)



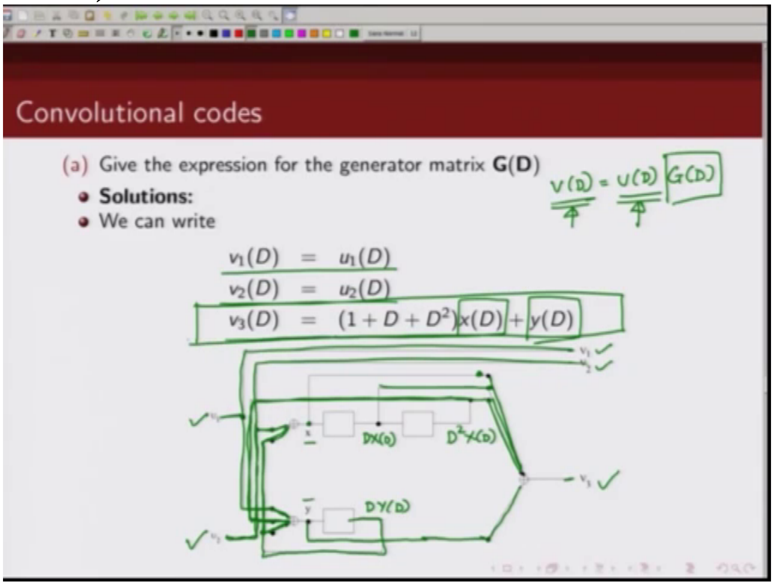
this one, Ok. So now we have got equations of y of D, x of D in terms of u 1 D and u 2 D. So let's write, bring y of D at one side and x of D at one side and write them in terms of, y of D and x of D in terms of u 1 D and u 2 D. So if we solve this, what we get is y of D is given by this and x of D is given by this. Now we plug these values of y of D and x of D given by this into

(Refer Slide Time 21:46)



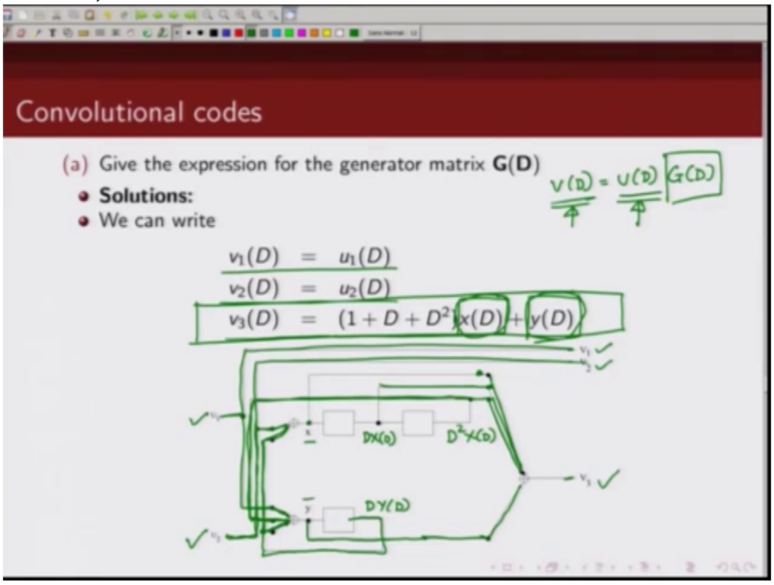
here, into this expression of

(Refer Slide Time 21:51)



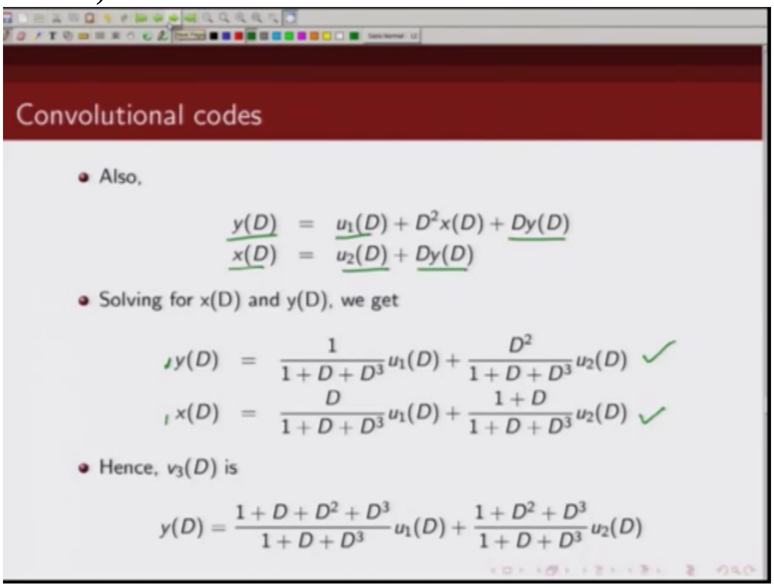
y 3 of D. So we plug this value of x D and y D which we just computed,

(Refer Slide Time 21:58)



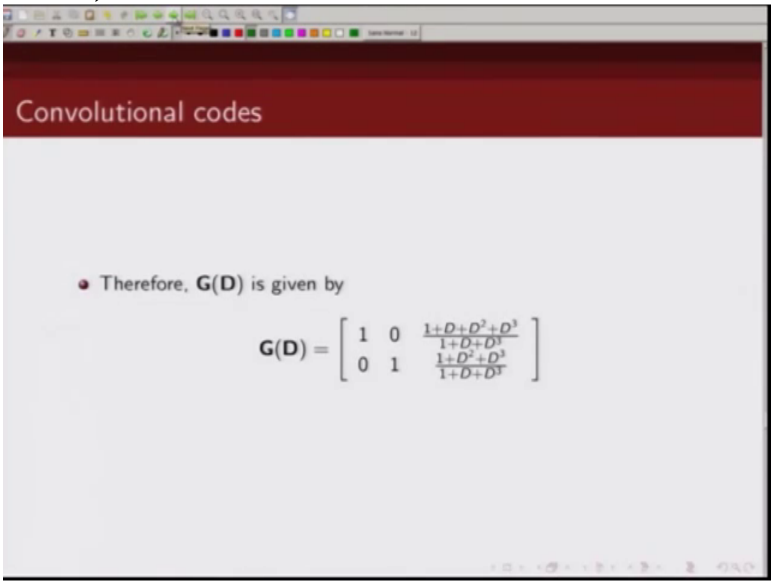
we plug those values in here. If we do that, we

(Refer Slide Time 22:03)



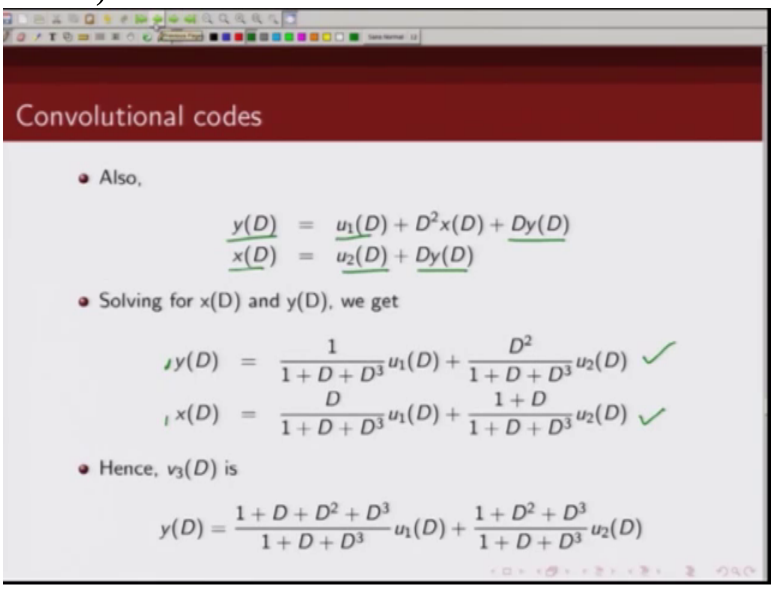
will get the expression of y 3 of D, Ok. Now,

(Refer Slide Time 22:11)



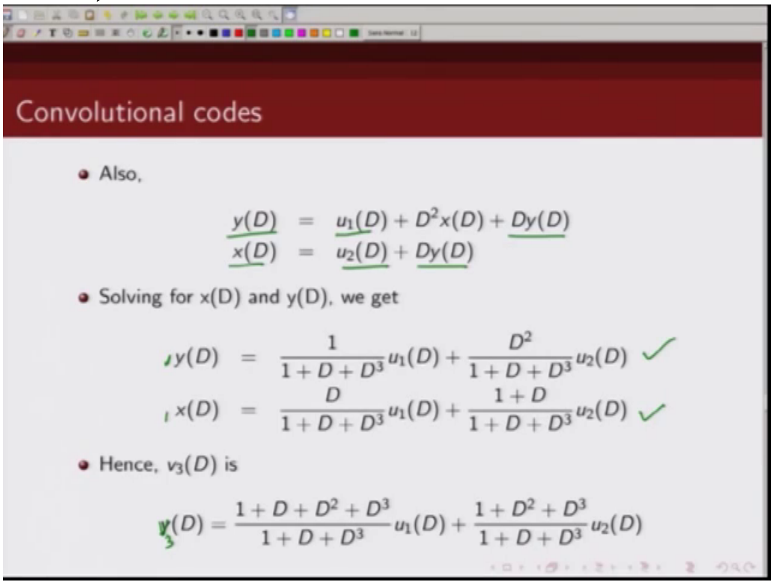
so if we do that finally

(Refer Slide Time 22:14)



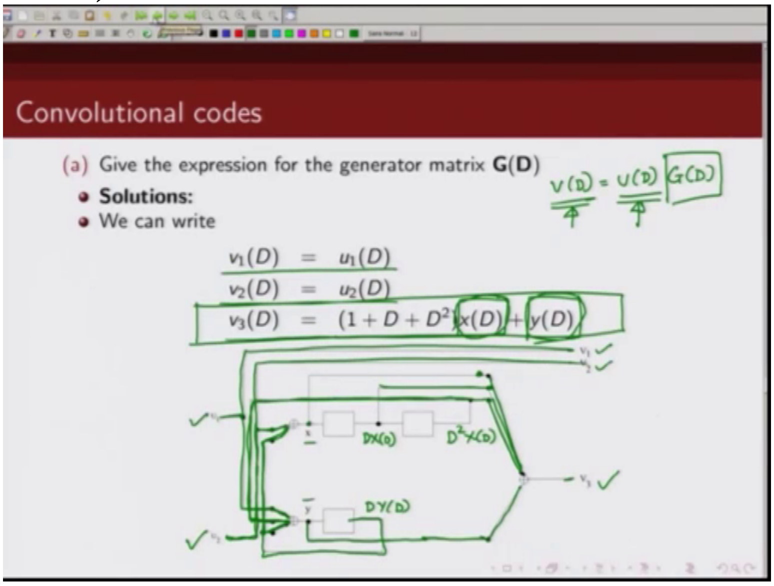
this is v 3 of D, so if we do that what we get is then v 3

(Refer Slide Time 22:21)



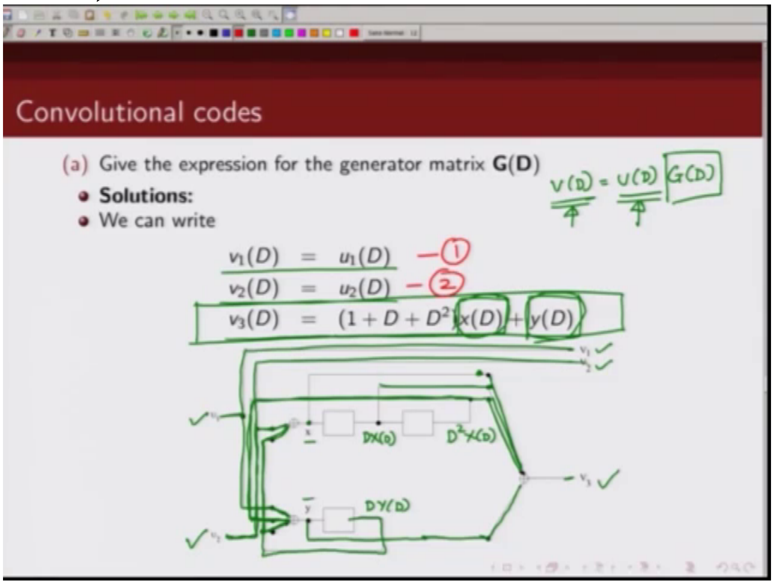
of D is this times u 1 D plus this times u 2 of D. So now we are in a position to

(Refer Slide Time 22:30)



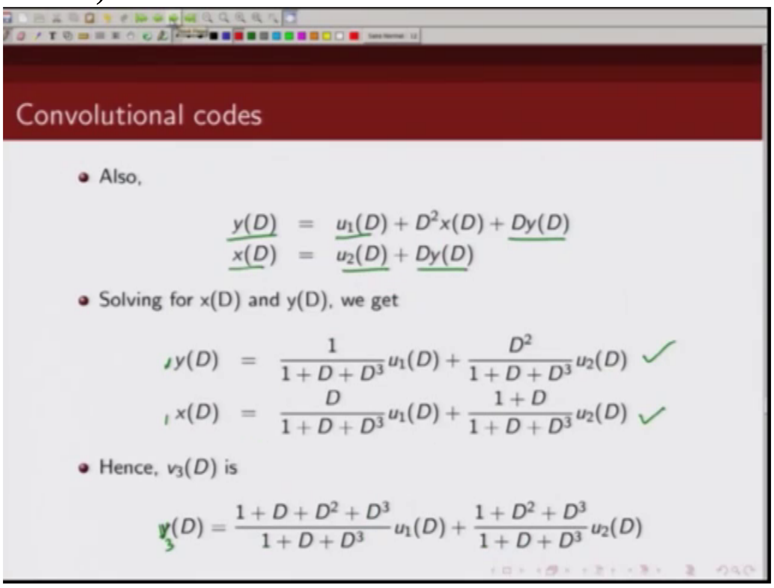
write the generator matrix. The first equation that we will require is this one. Second equation we will require is this one.

(Refer Slide Time 22:39)



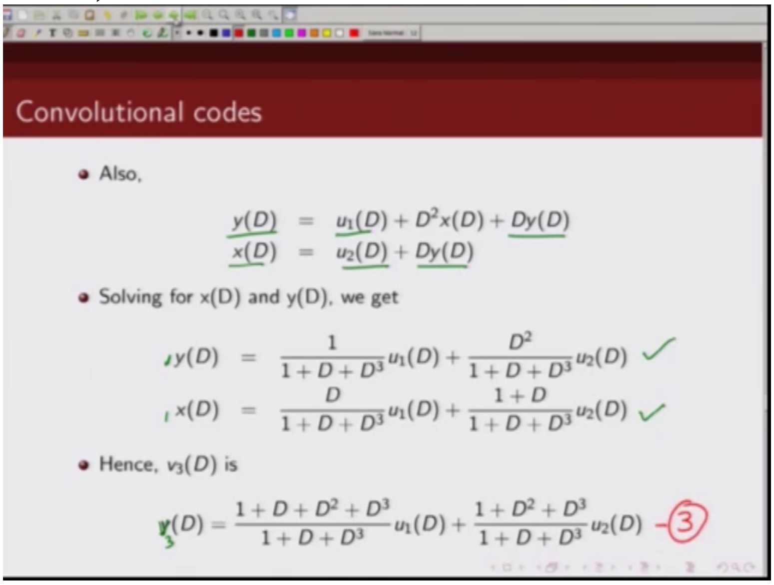
And the third

(Refer Slide Time 22:40)



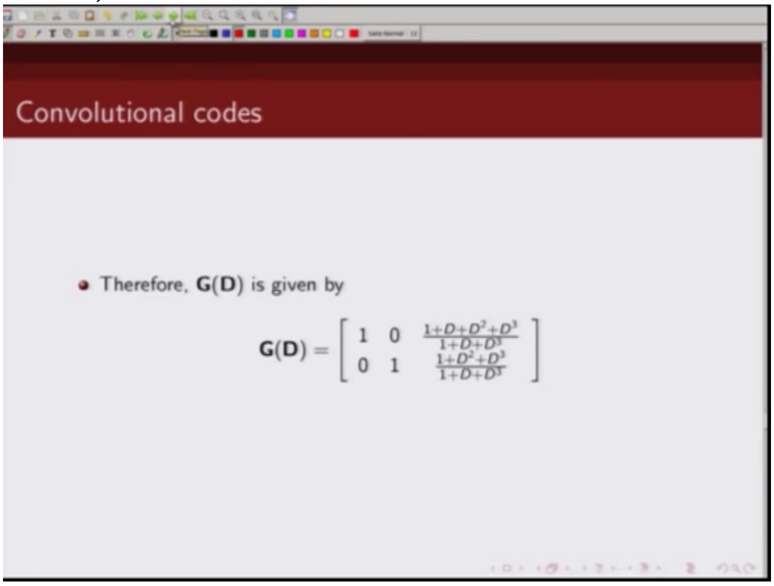
equation that we will require is this one,

(Refer Slide Time 22:43)



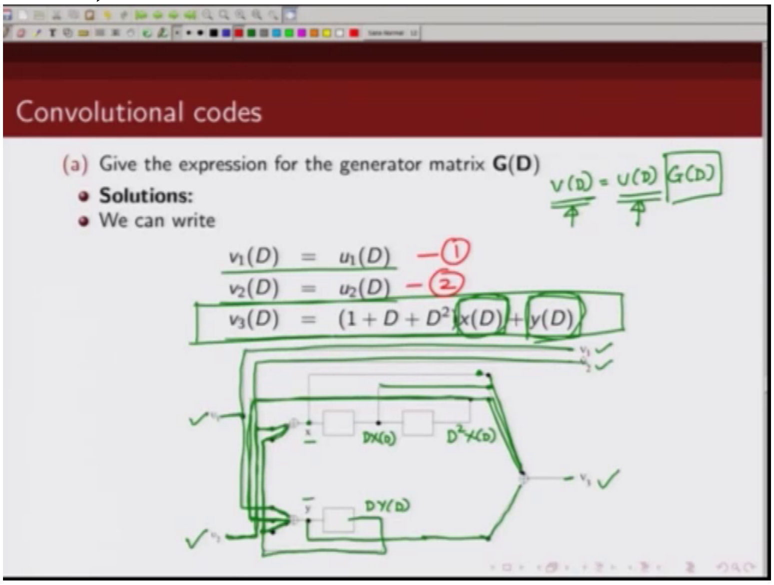
right. So

(Refer Slide Time 22:44)



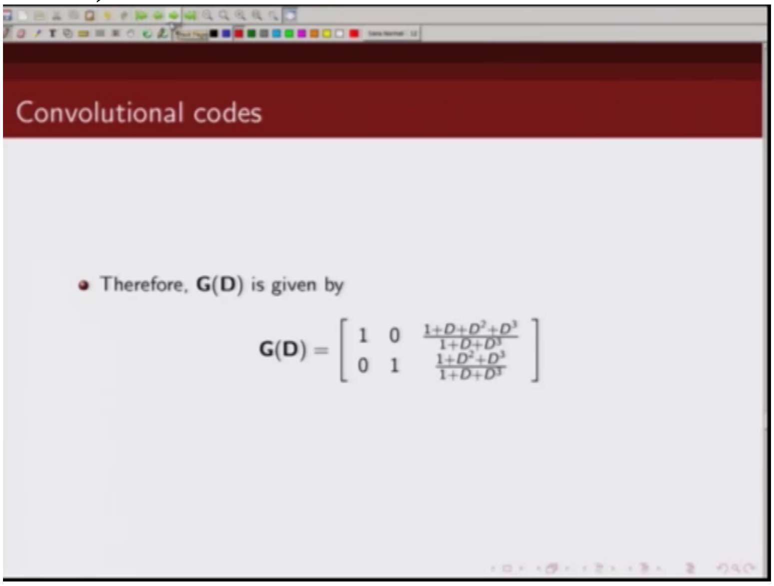
you can think of it as like this, so we have 3

(Refer Slide Time 22:49)



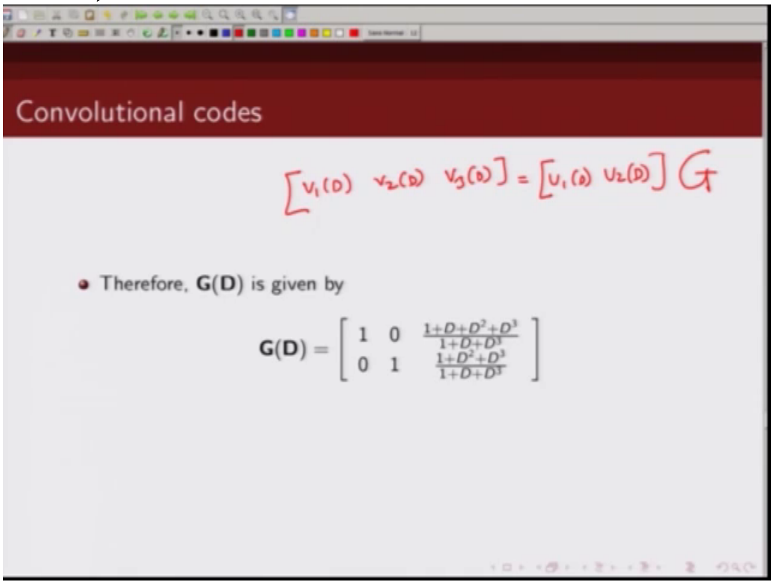
output, v 1, v 2, v 3, 2 input u 1, u 2, so we are writing

(Refer Slide Time 22:56)



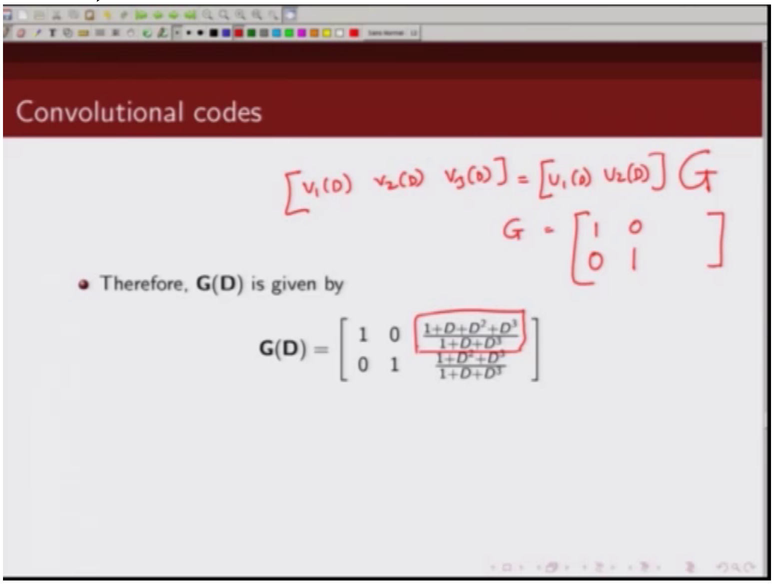
v 1 D, v 2 D, v 3 D in terms of u 1 D, u 2 D and this G matrix.

(Refer Slide Time 23:13)



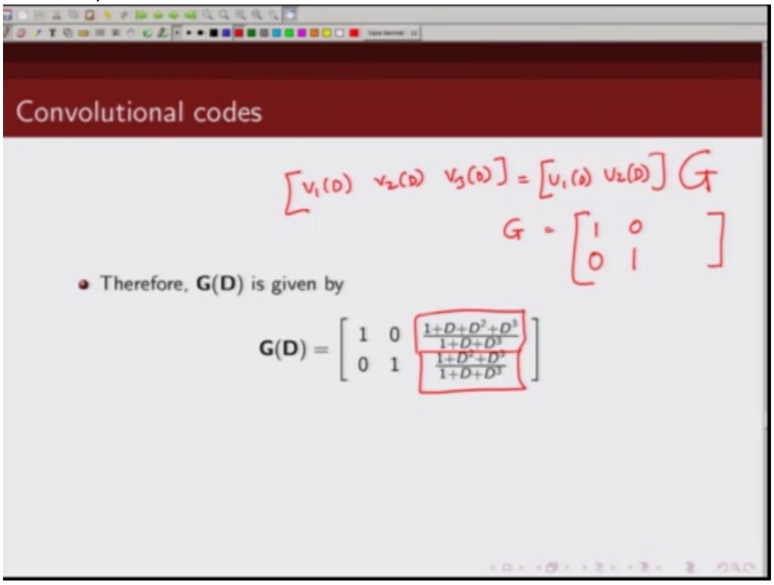
So what is v 1 D? v 1 D is u 1 of D. So then our G matrix here, again G matrix is 2 cross 3, so v 1 D is u 1 of D, so we get 1 0. v 2 of D is u 2 D, so we get 0 1, and what is v 3 of D, v 3 of D is this, this times

(Refer Slide Time 23:42)



u 1 of D and this times u 2 of D.

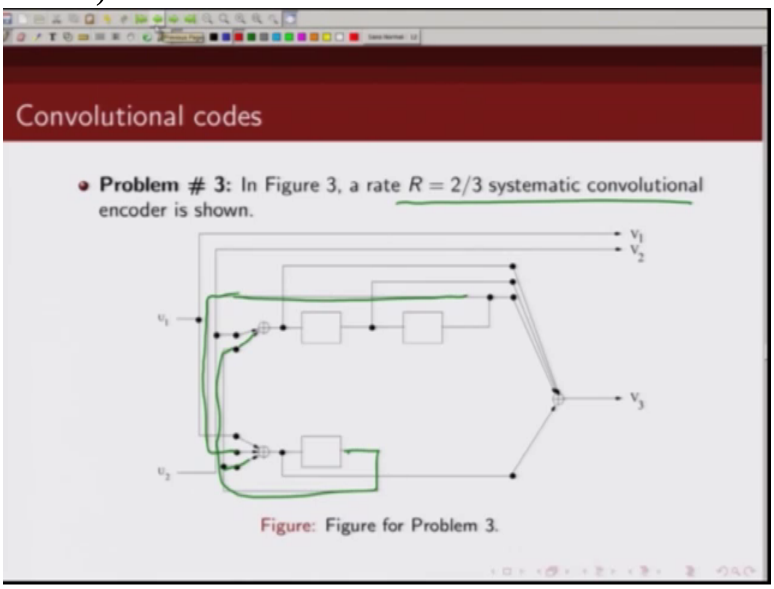
(Refer Slide Time 23:46)



$$G(D) = \begin{bmatrix} 1 & 0 & \frac{1+D+D^2+D^3}{1+D+D^3} \\ 0 & 1 & \frac{1+D^2+D^3}{1+D+D^3} \end{bmatrix}$$

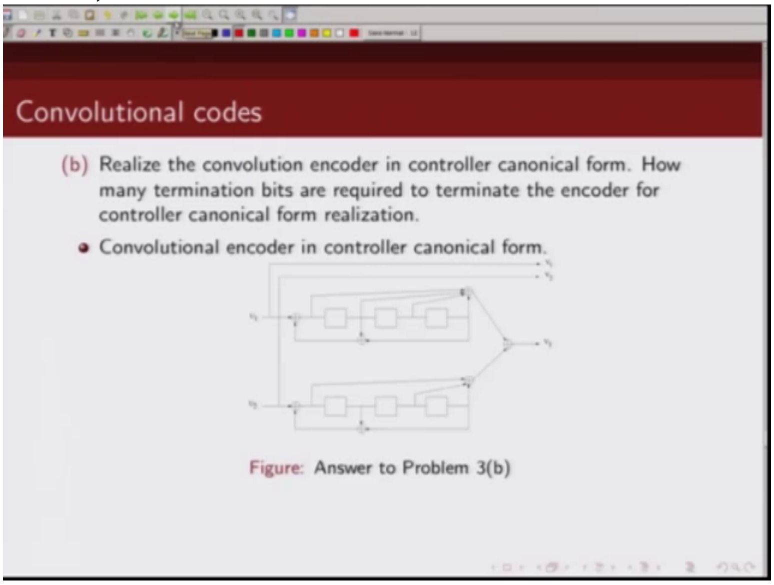
So this will be our final generator matrix corresponding to the encoder that is shown in this

(Refer Slide Time 23:58)



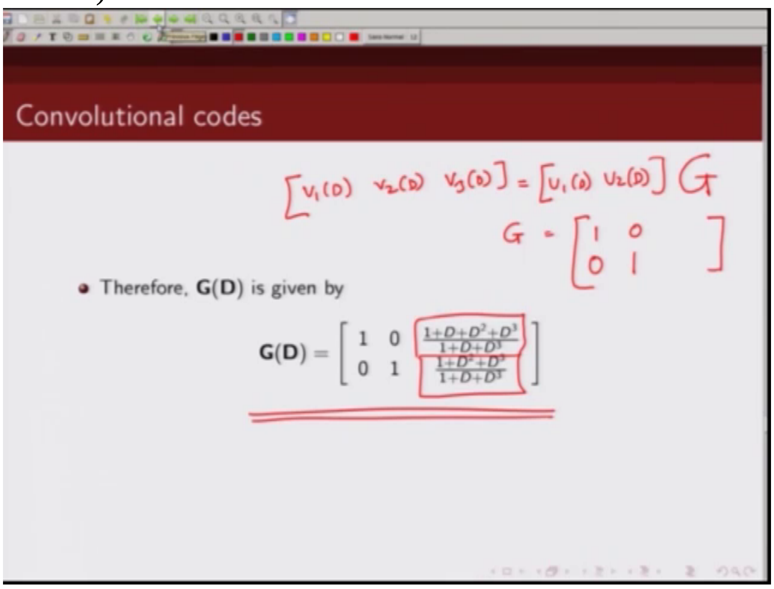
figure, Ok. Now

(Refer Slide Time 24:03)



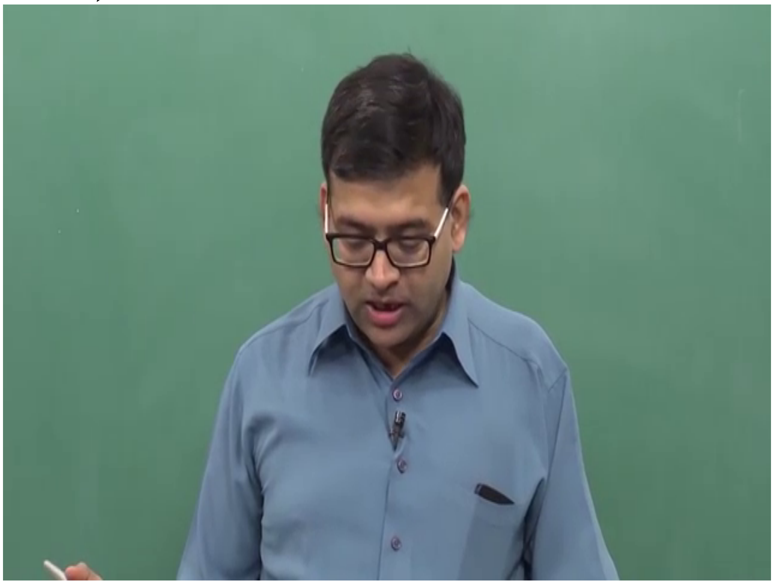
the next question is can we realize this encoder in the controller canonical form? So the answer is yes, we can realize it.

(Refer Slide Time 24:14)



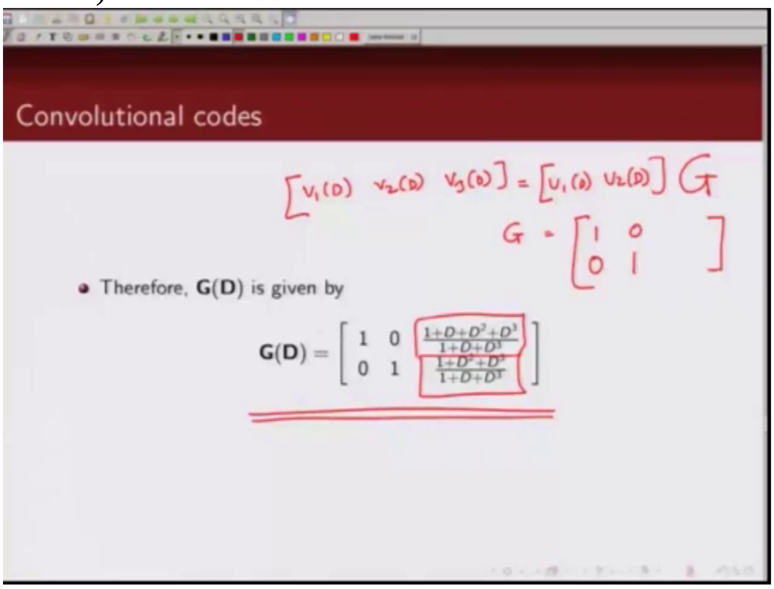
We can have the expression for generator matrix. So to realize it in controller

(Refer Slide Time 24:22)



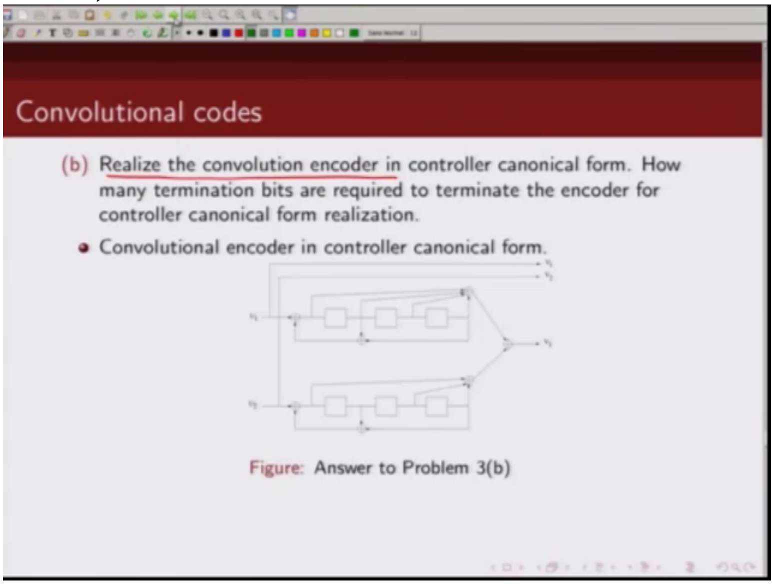
canonical form again, so there is one

(Refer Slide Time 24:25)



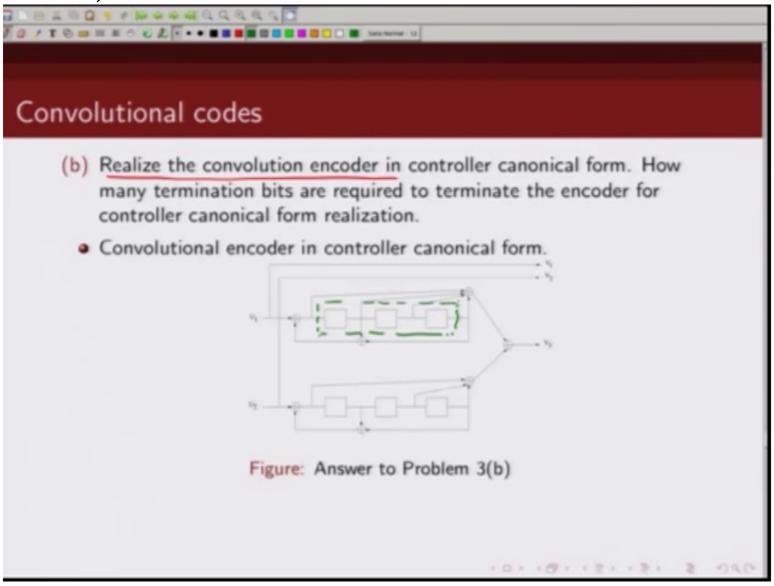
set of shift registers for each input, so this is one input, this is second input right. Please note this is a feedback polynomial so we would require a feedback polynomial and now maximum degree here is 3, maximum degree here also is 3,so

(Refer Slide Time 24:46)



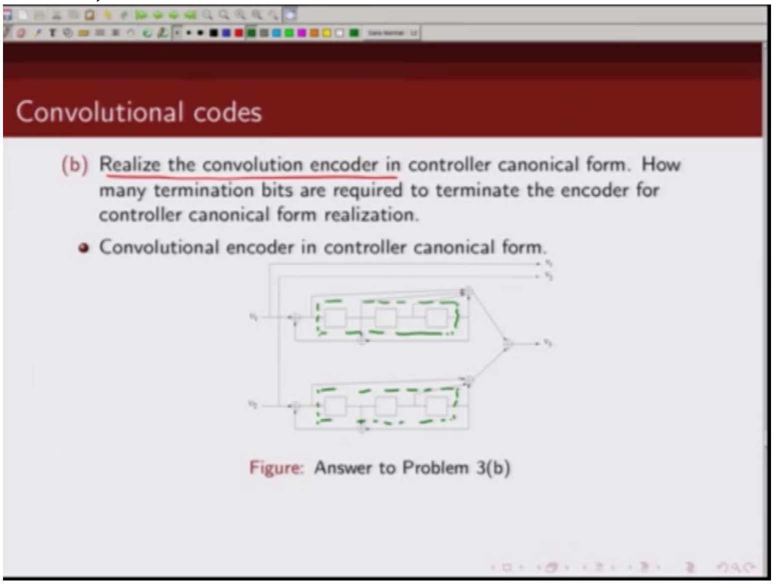
we will require 2 set of shift registers, first one is this one. Please note this as

(Refer Slide Time 24:58)



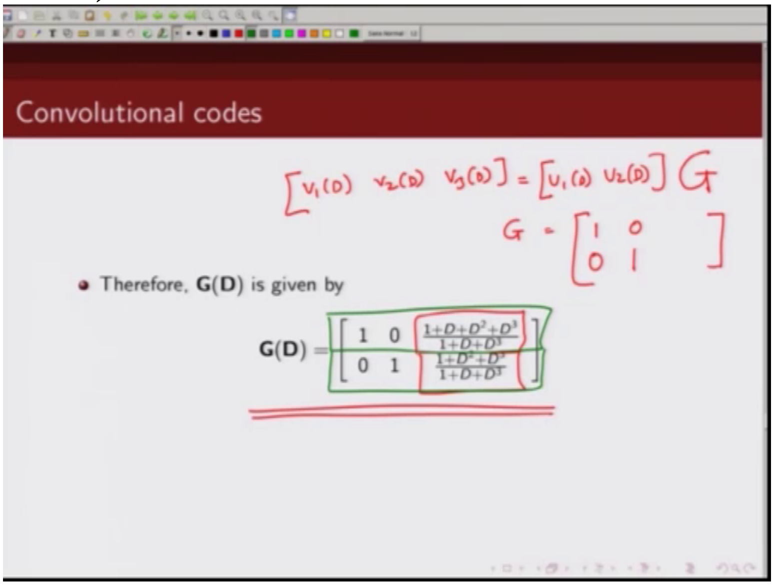
3 memory elements and similarly second shift register, this also has 3 memory elements.

(Refer Slide Time 25:05)



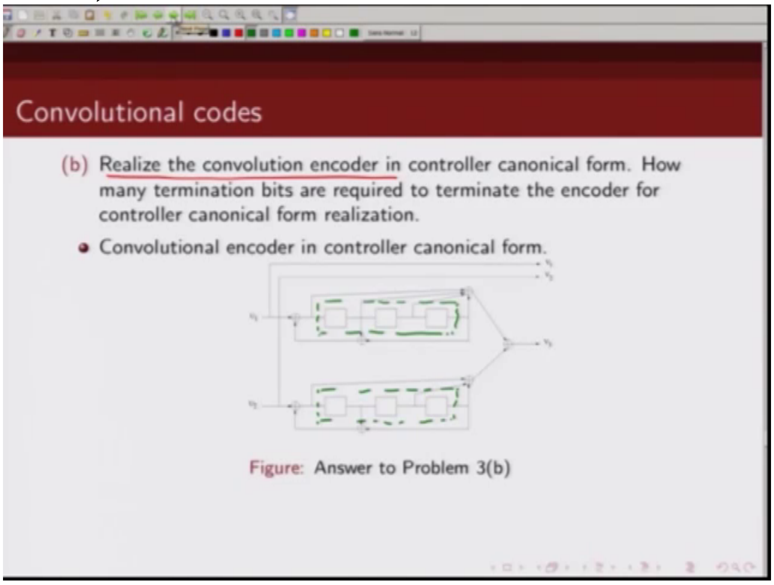
That is because the maximum degree

(Refer Slide Time 25:08)



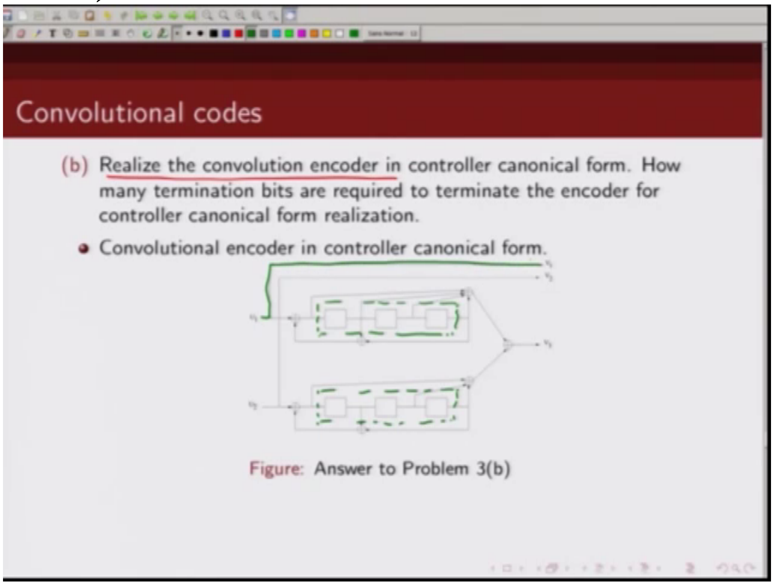
of this rational function is 3 and similarly maximum degree of this rational function is 3. And we just implement this.

(Refer Slide Time 25:20)



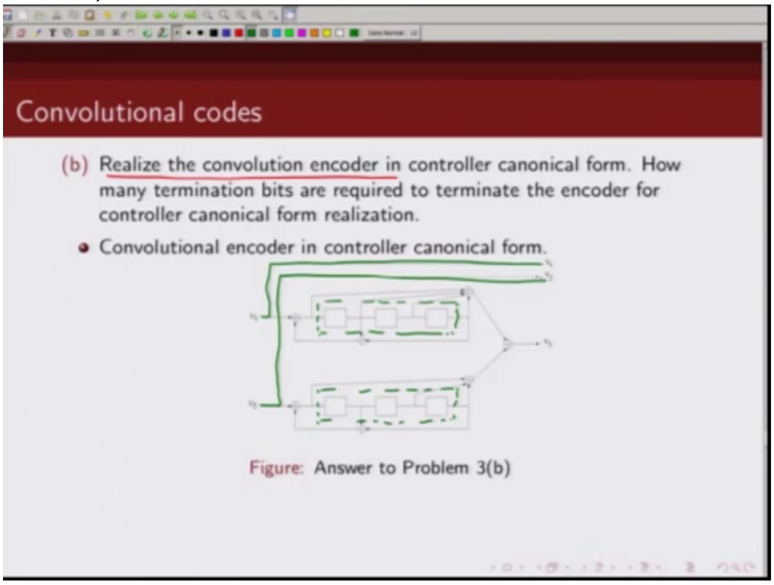
So v 1, v 1 D is just u 1 D, so that's just this.

(Refer Slide Time 25:26)



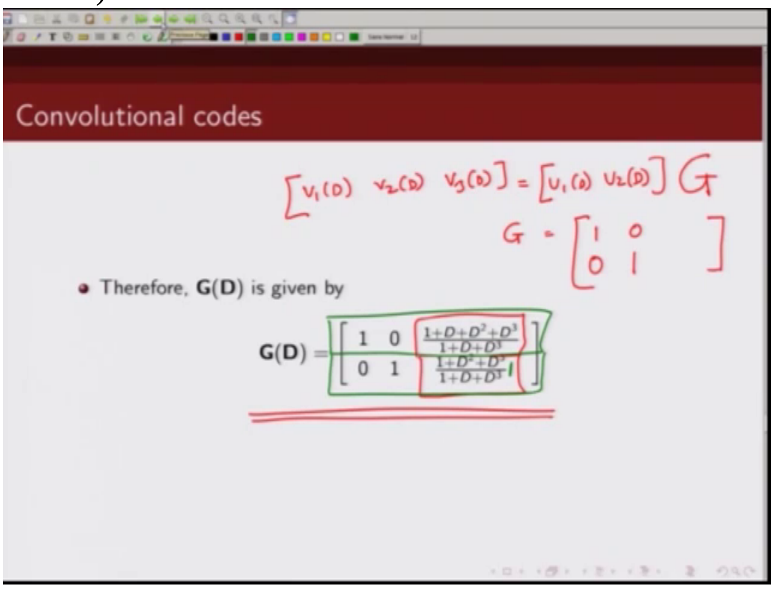
v 2 D is u 2 D, that's just this.

(Refer Slide Time 25:32)



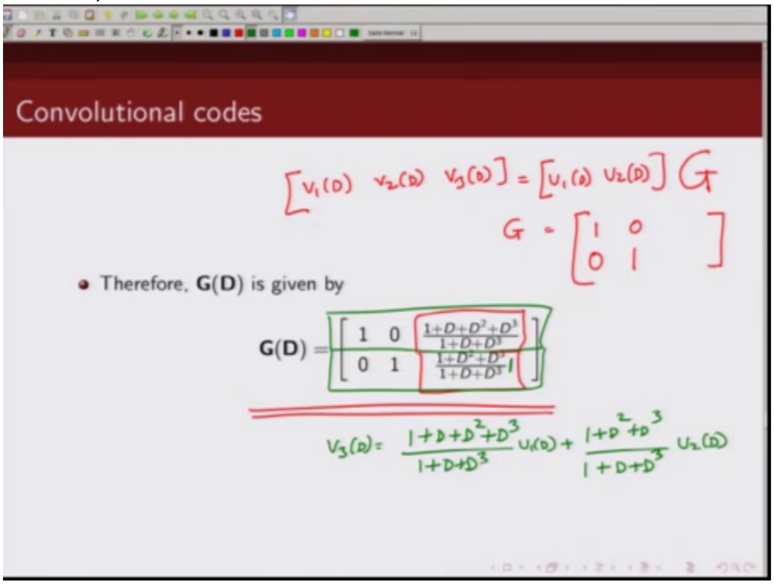
What is

(Refer Slide Time 25:35)



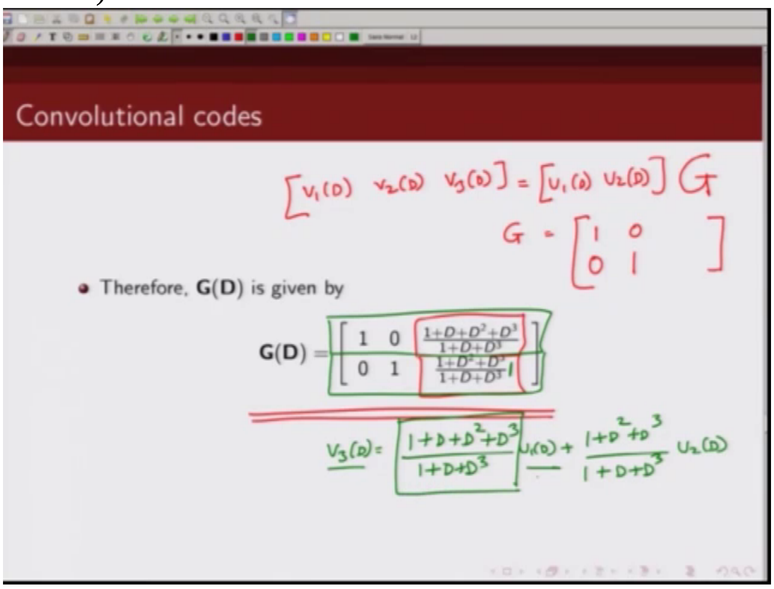
v 3 D? v 3 D is 1 plus D plus D square plus D cube divided by 1 plus D plus D 3 u 1 D, plus 1 plus D square plus D cube 1 plus D plus D cube u 2 D,

(Refer Slide Time 25:59)



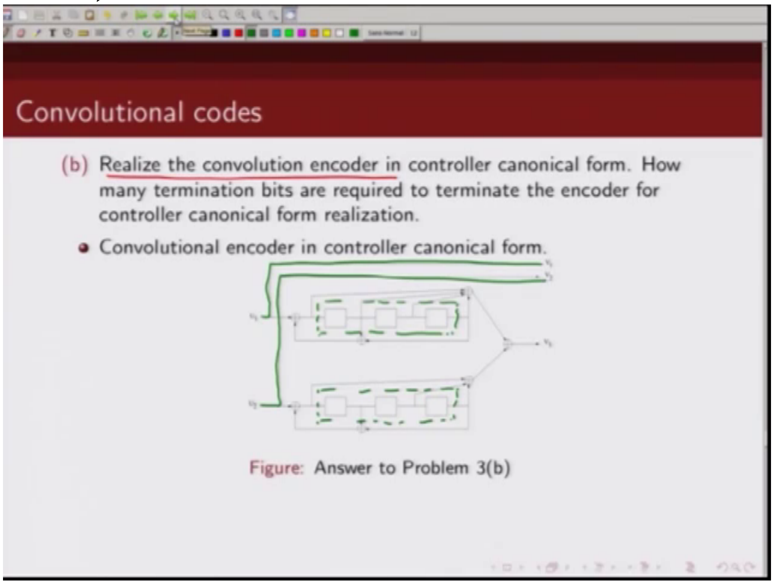
right? So relationship between v 3 and u 1 D is given by this. So let's

(Refer Slide Time 26:07)



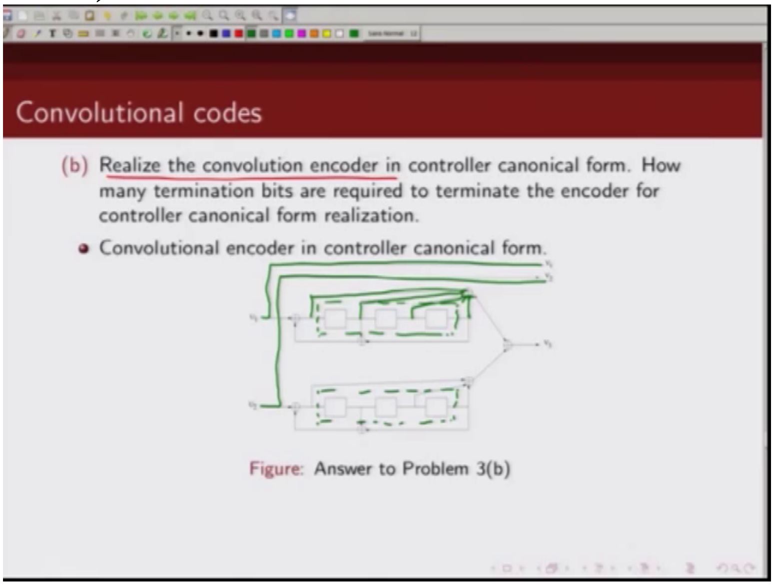
implement this. So numerator has 1 plus D plus D square plus D cube. So you can see here,

(Refer Slide Time 26:14)



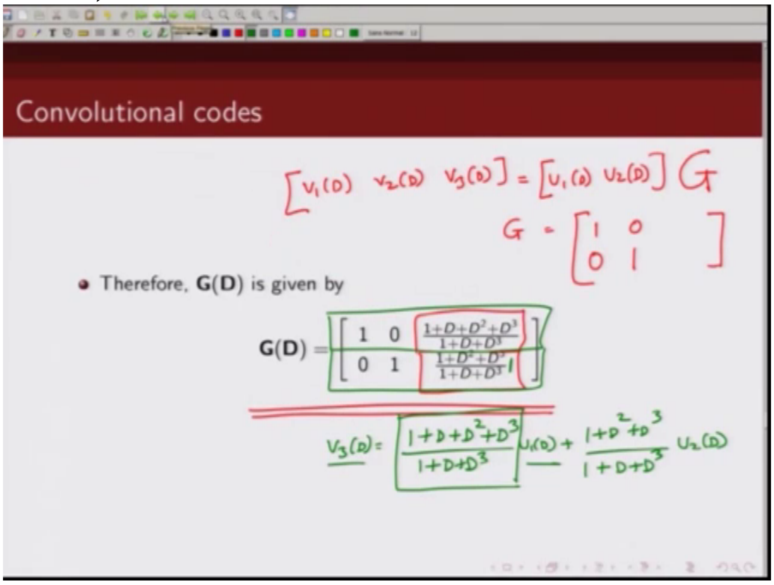
this is my 1, this is my D, this is my D square, this is my D cube.

(Refer Slide Time 26:24)



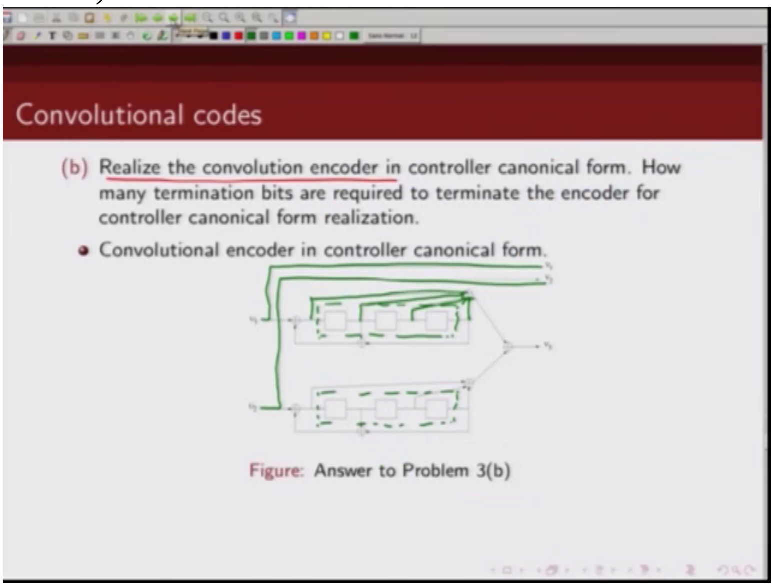
And similarly the denominator has 1

(Refer Slide Time 26:26)



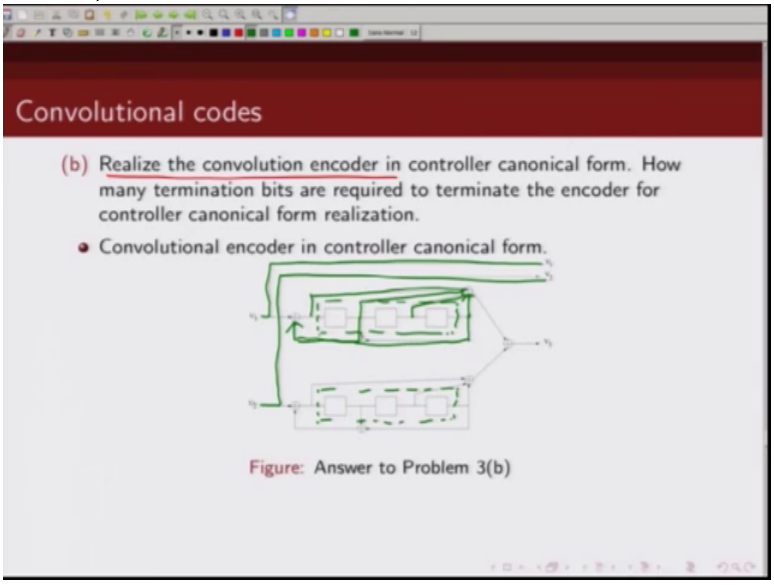
plus D plus D cube. So

(Refer Slide Time 26:31)



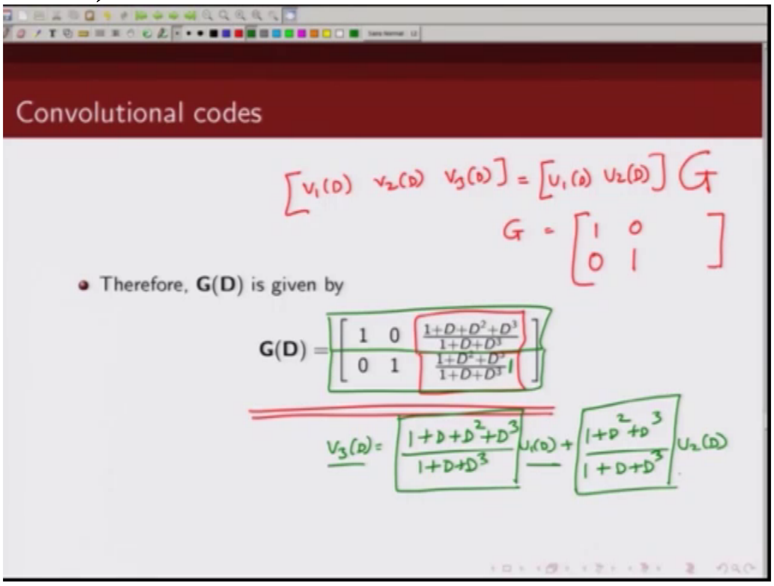
the denominator, this is the 1 term, this is the D term; this is the D cube term. So this part is implemented.

(Refer Slide Time 26:43)



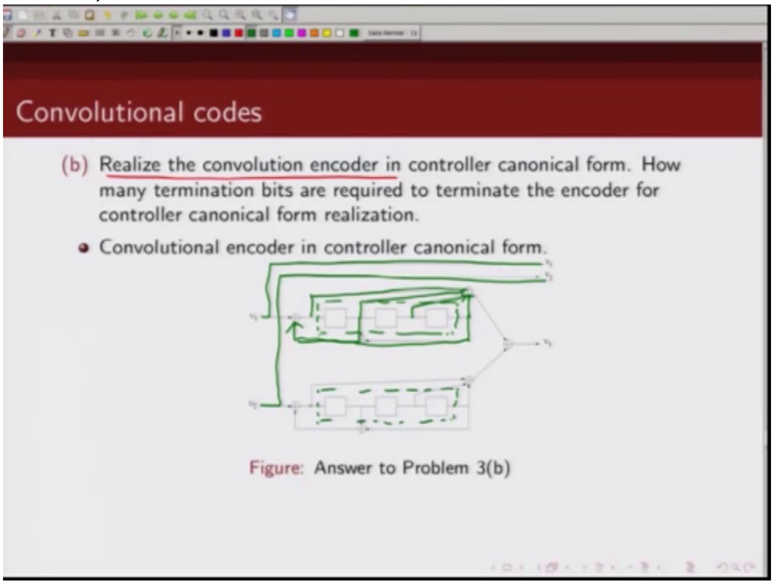
Next is this. Following the same procedure

(Refer Slide Time 26:52)



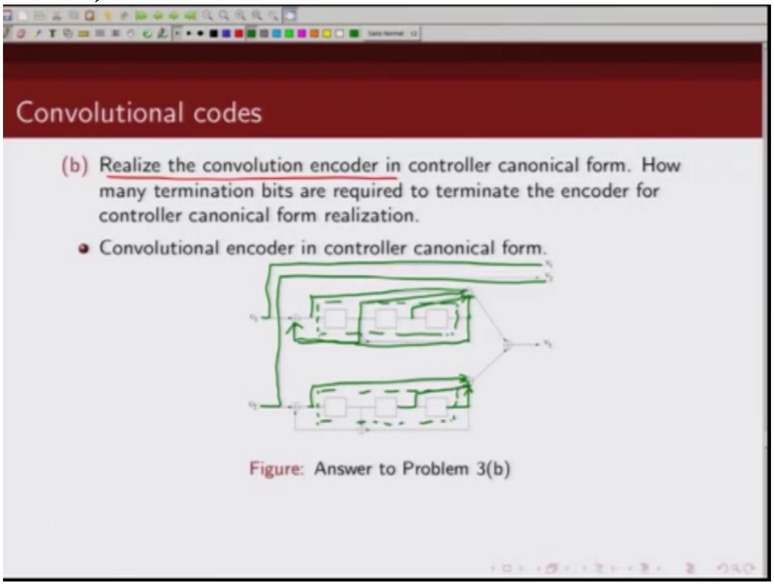
we can find out the mapping between u 2 D and v 3 D. The feed forward connections are 1, D square and D cube. So then this is

(Refer Slide Time 27:03)

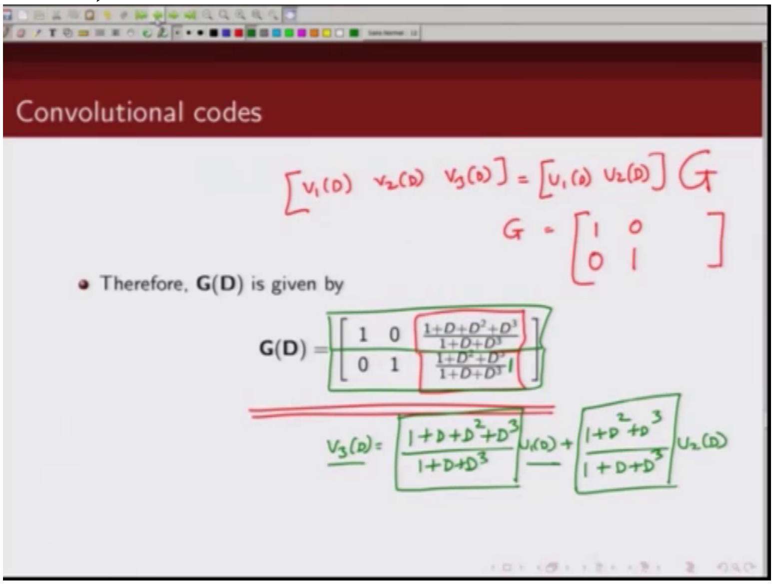


1, this is D no connection,D square is this and D cube is this.

(Refer Slide Time 27:11)

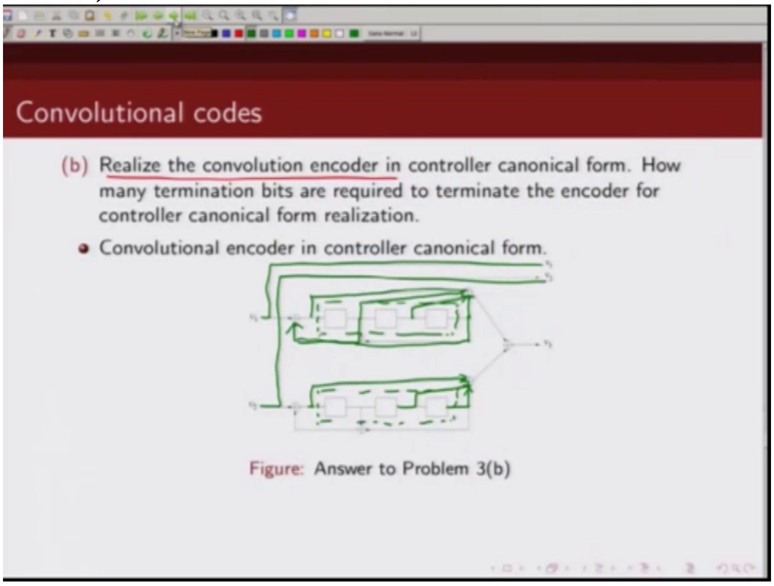


(Refer Slide Time 27:12)



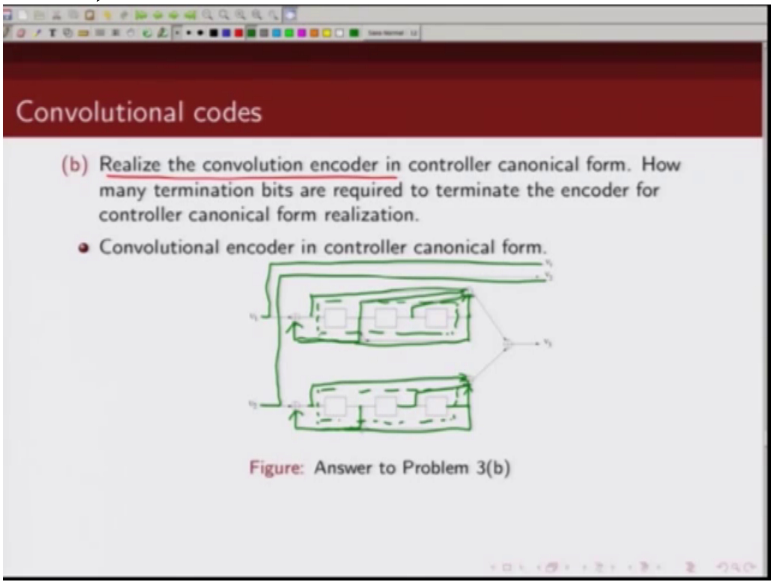
Similarly the feedback connections are 1, D and D cube. So

(Refer Slide Time 27:18)



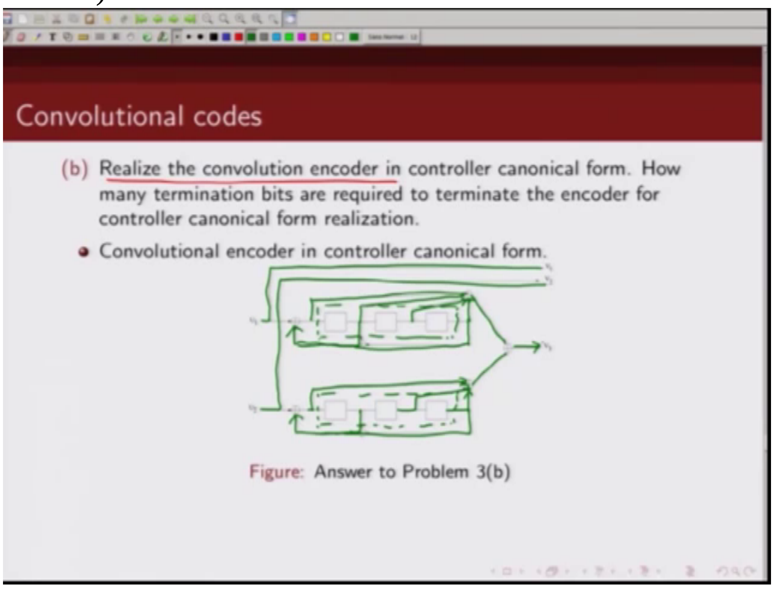
the feedback connections; this is the 1, this is D and this is D cube. And

(Refer Slide Time 27:28)



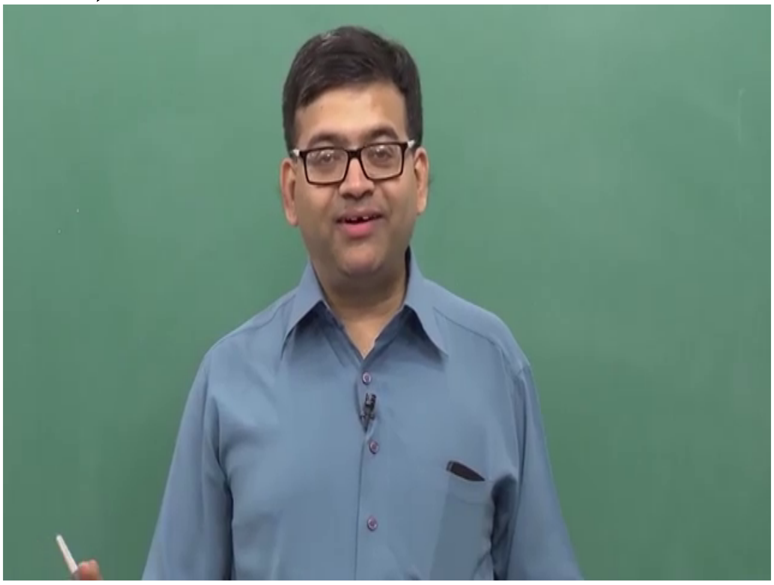
v 3 is the combination of these 2. So this is my v 3.

(Refer Slide Time 27:33)



So I hope this is clear how we can realize

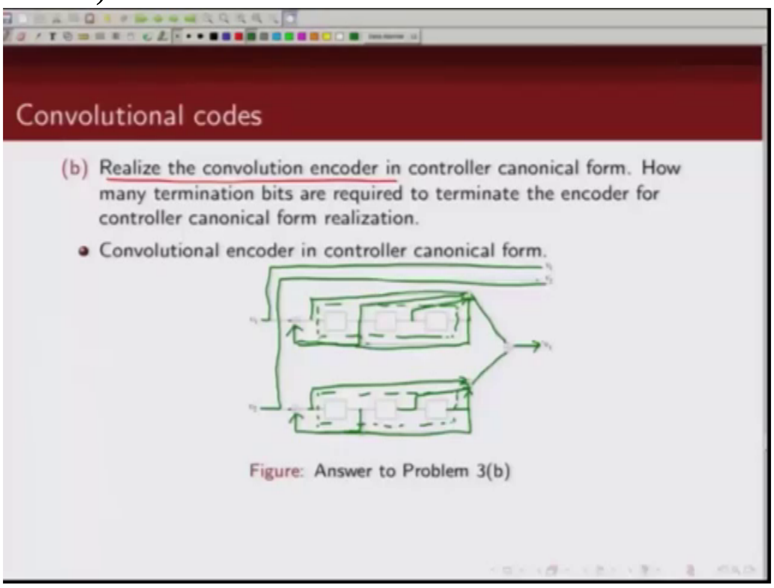
(Refer Slide Time 27:39)



this encoder using controller canonical form realization.

Now the next

(Refer Slide Time 27:45)

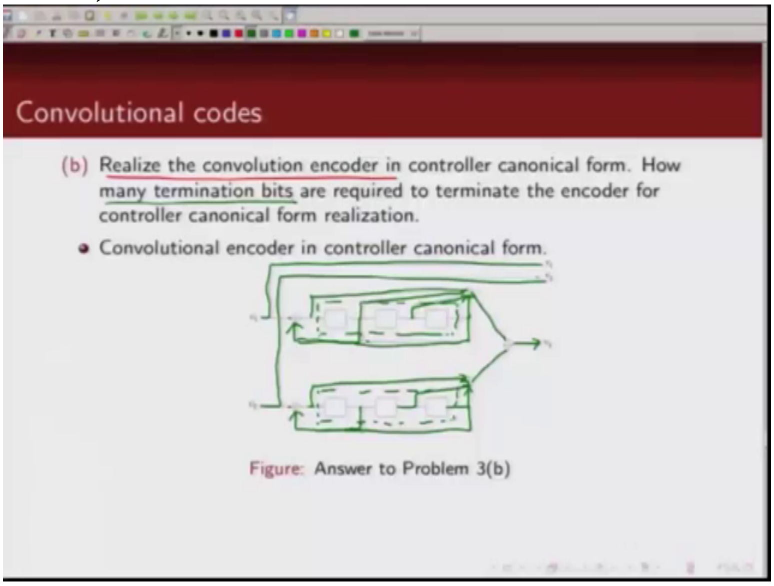


question is how many termination bits are required to bring this encoder back to all zero state? Now what does termination means? Termination means

(Refer Slide Time 27:55)

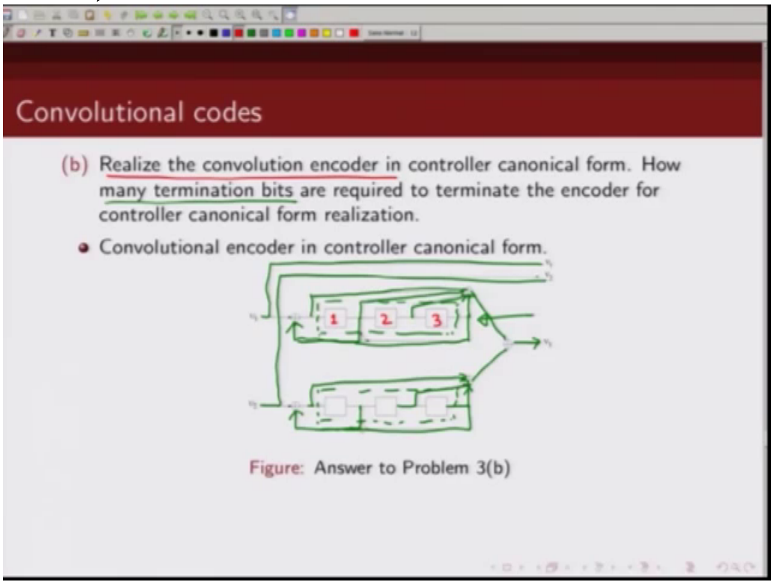
we are bringing this encoder back to all zero state. So no matter what the state is, if you want to bring them back

(Refer Slide Time 28:06)



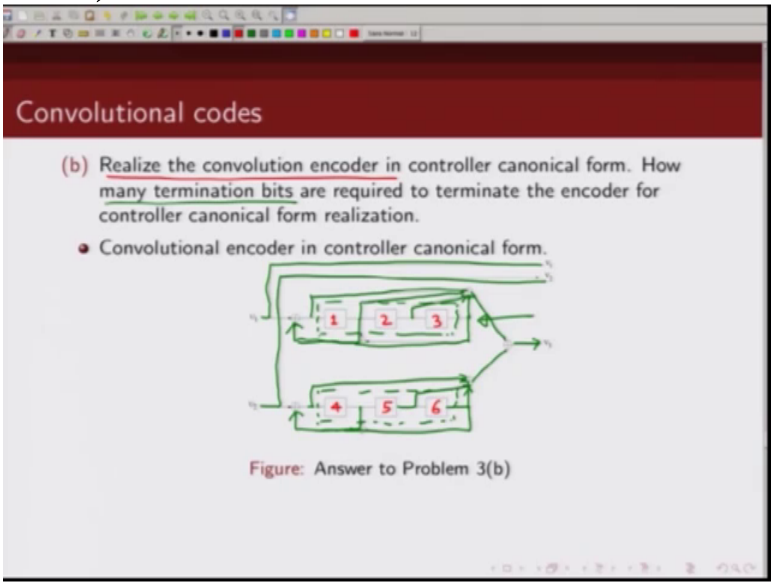
to the all zero state, the number of termination bits required is equal to how many memory elements we have. So in the controller canonical form realization to bring this shift register, the first shift register you want to bring it to all zero state, we would require 3 bits because we have 3 memory elements here, 1, 2 and 3. Similarly

(Refer Slide Time 28:32)



for this shift register we require additional 3 bits, so 4, 5, 6 so

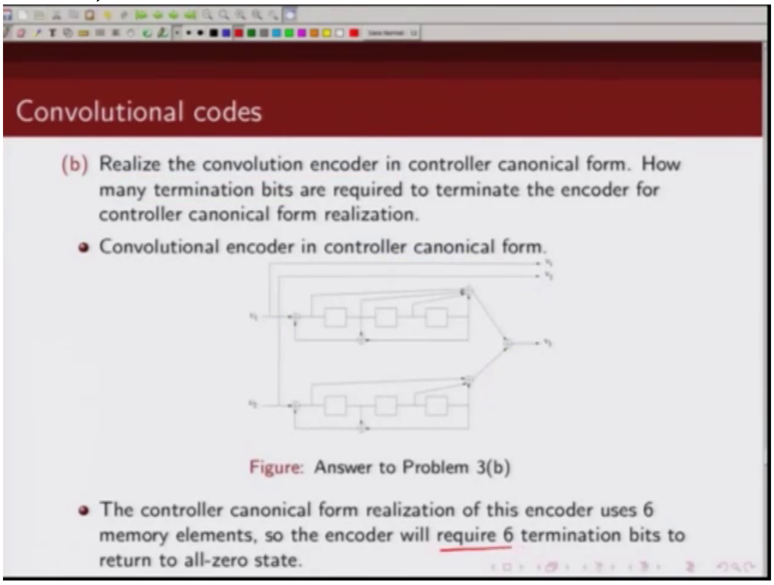
(Refer Slide Time 28:40)



total we require 6 termination bits, 3 to terminate this encoder and 3 to terminate this encoder.

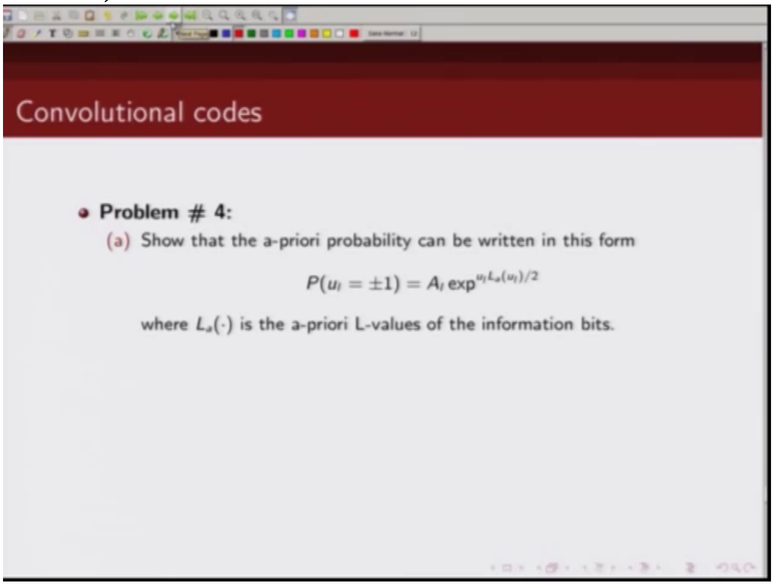
So we require

(Refer Slide Time 28:51)



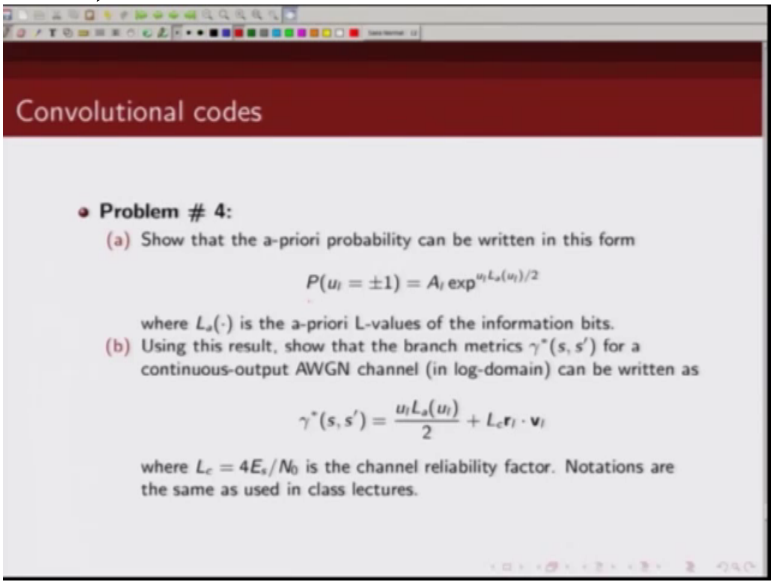
6 termination bits.

(Refer Slide Time 28:54)



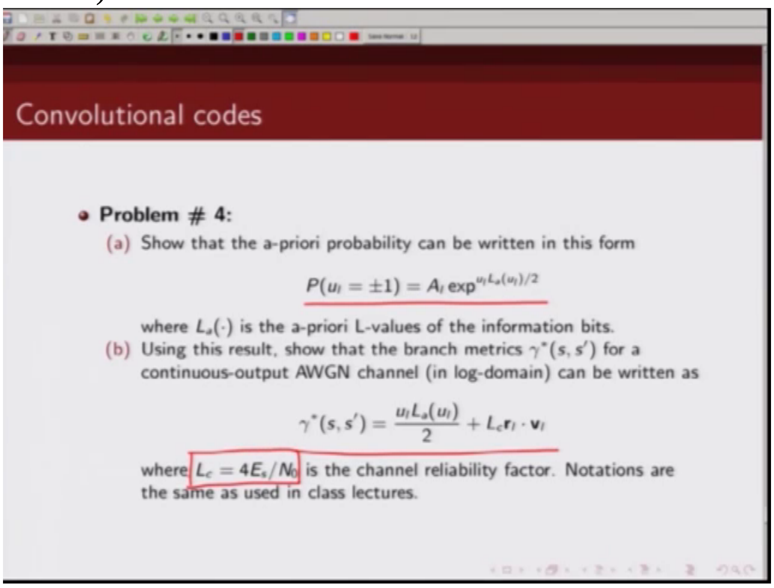
Finally let's come to the B C J R algorithm that we talked about. So the first question is can you write

(Refer Slide Time 29:03)



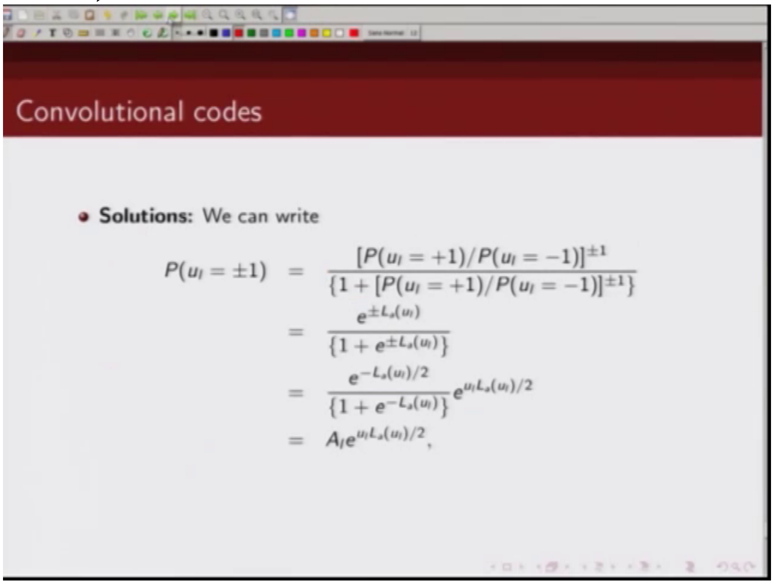
the a priori probability in this particular form, and also the branch metric in log domain, can it be written in this particular form? Now u l is my input, L a is the a p p value for the a priori inputs, L c is the reliability factor which is given by 4 times E s by N naught,

(Refer Slide Time 29:28)



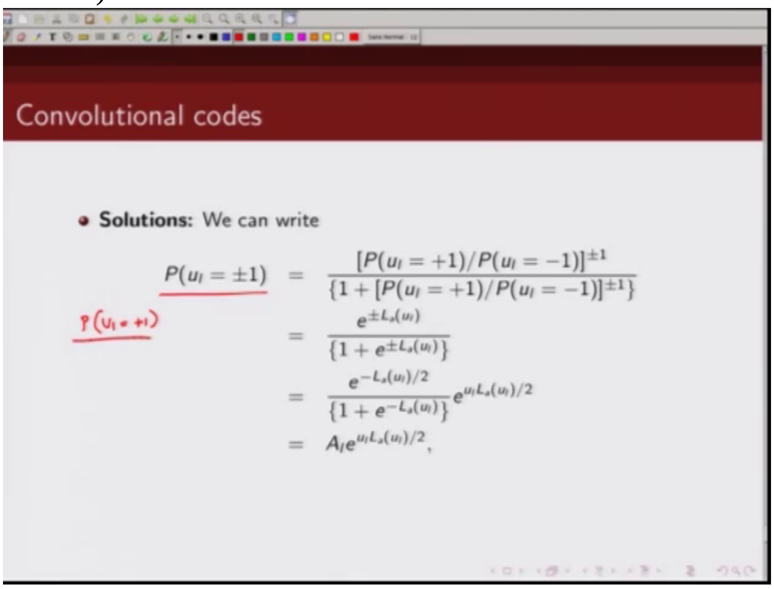
other notations are same as which are used in the lecture. v is codeword, r is received sequence so can we write these in terms like this. So let's look at it.

(Refer Slide Time 29:46)



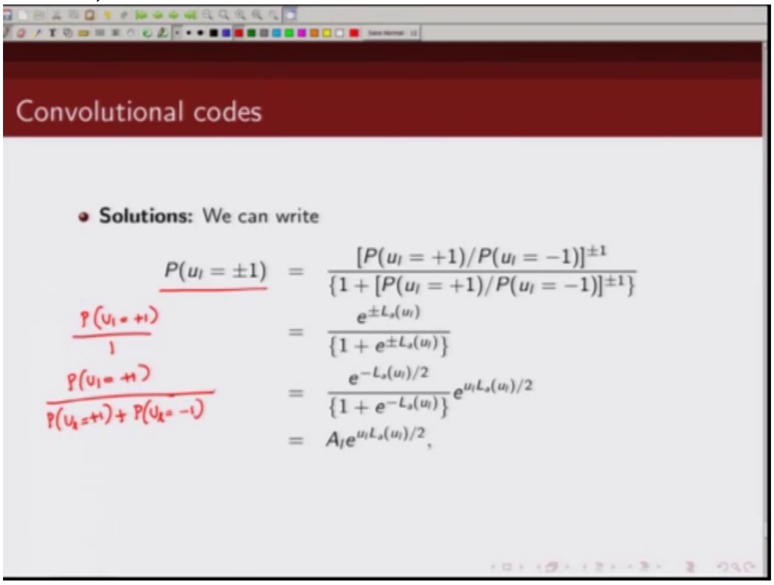
So what's the probability of u being plus 1 or minus 1? Let's take like plus 1, let's say what's the prob, what's the probability that u l is plus 1? Now this can be written as this by 1.

(Refer Slide Time 29:59)



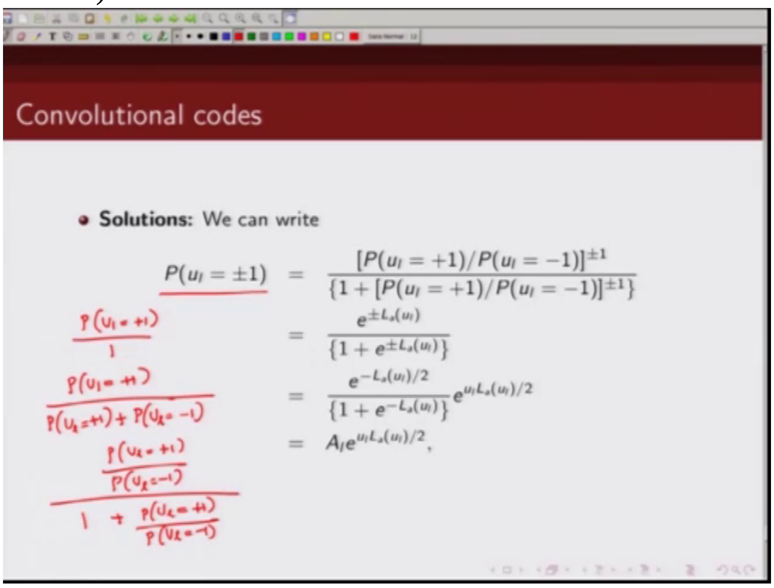
So I can write as probability of u l being plus 1 divided by probability u l is plus 1 plus probability u l is minus 1, I can write it

(Refer Slide Time 30:13)



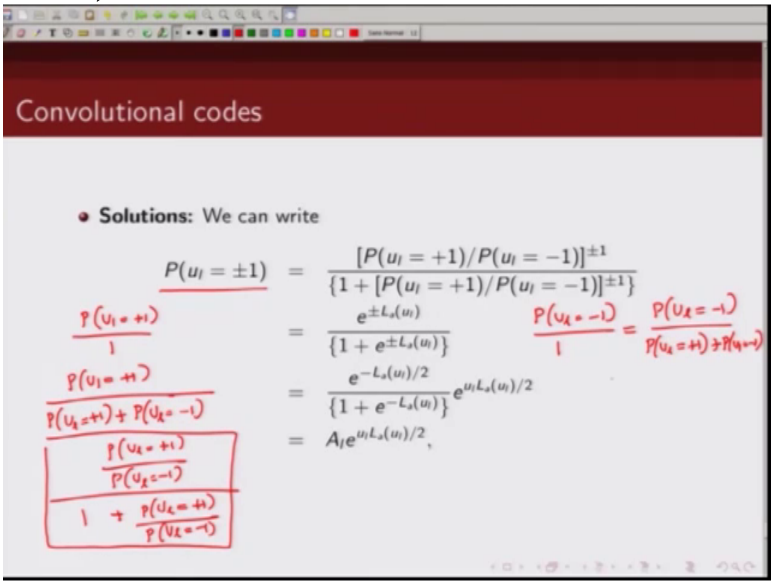
this way, right? And if I divide by probability of u l being minus 1, then what I get is probability being plus 1 by probability of u l being minus 1, 1 plus probability of u l being plus 1, probability of u l minus 1.

(Refer Slide Time 30:44)



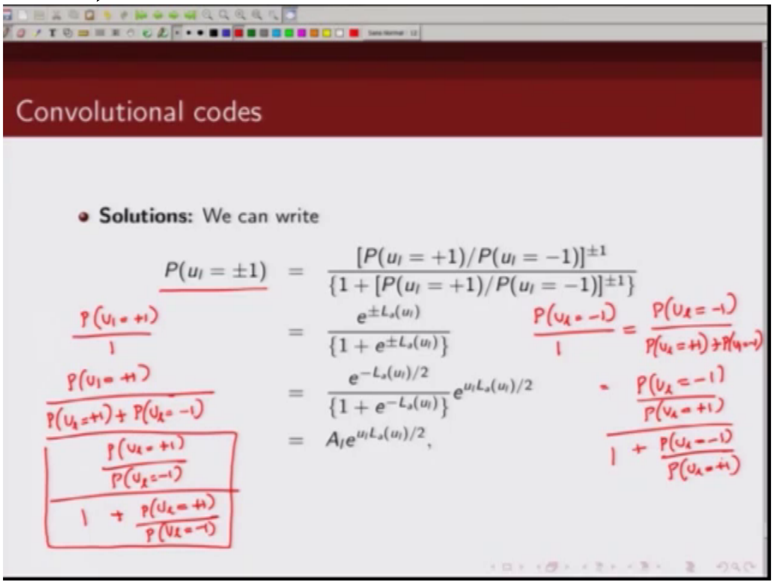
So this is what I will get of the form here. You can see here, the form for, when u l is plus 1, I get probability of u l being plus 1, I get in this particular form. Now let's look at what's the probability that u l is minus 1. Again I can follow the same procedure. I can write this same as this by 1 or I can write the probability of u l is equal to minus 1 divided by probability of u l being plus 1 plus probability of u l being minus 1 and

(Refer Slide Time 31:23)



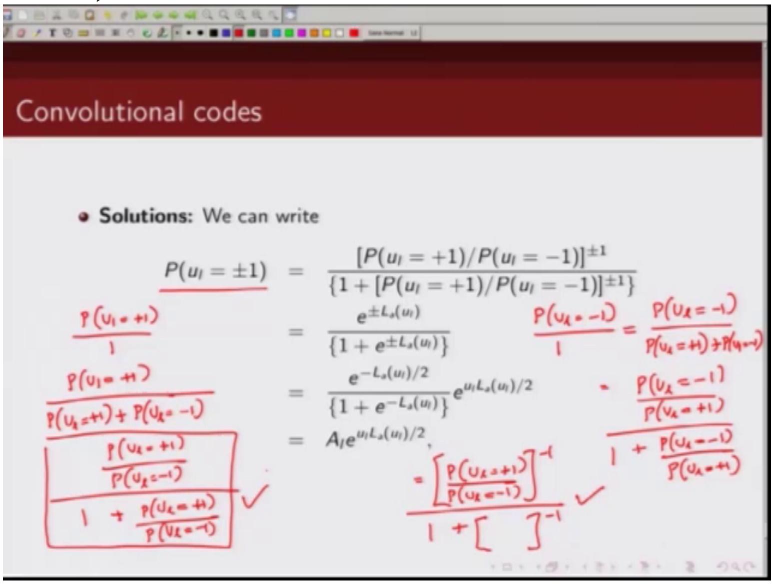
I can divide this by probability of u l being 1, so this will be probability of u l equal to minus 1 by probability of u l being plus 1, 1 plus probability of u l being minus 1, probability of u l being plus 1, right?

(Refer Slide Time 31:49)



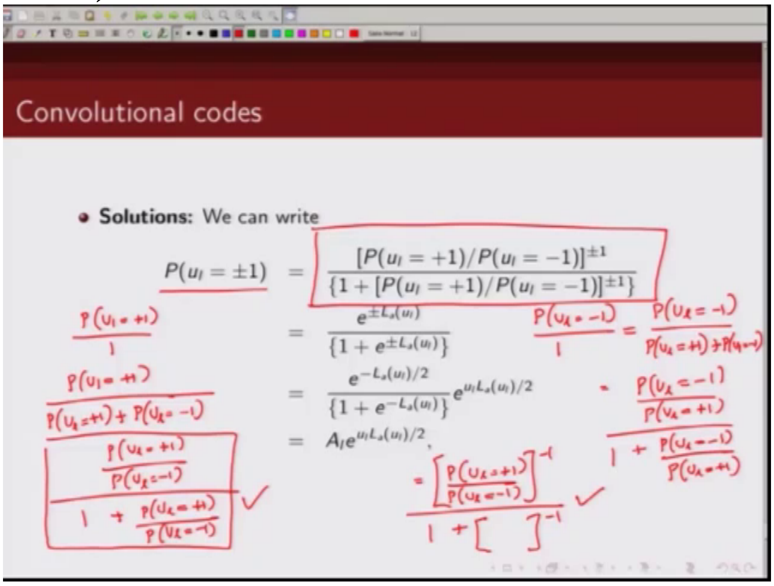
So this I can also write as this is equal to probability of u l being plus 1 by probability of u l being minus 1 raised to power minus 1 and this is 1 plus the same thing, raised to power minus 1. So if I combine this and this what I get

(Refer Slide Time 32:15)



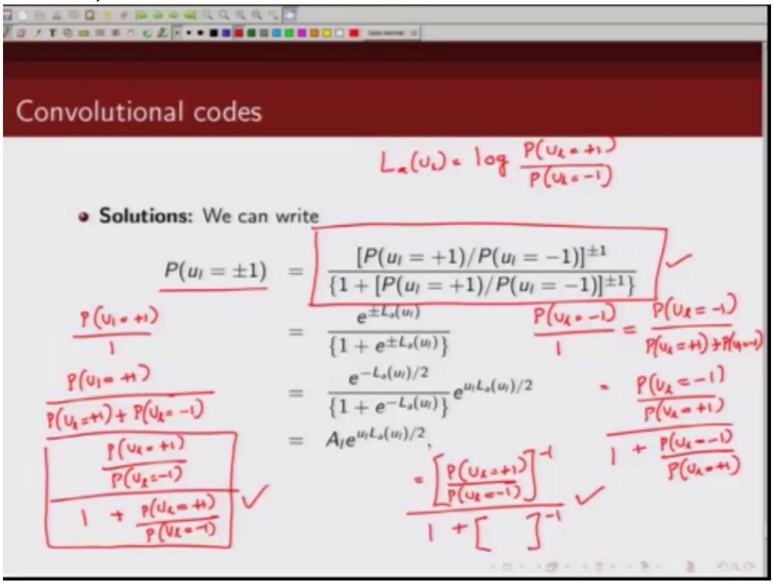
is the first step here, Ok.

(Refer Slide Time 32:20)



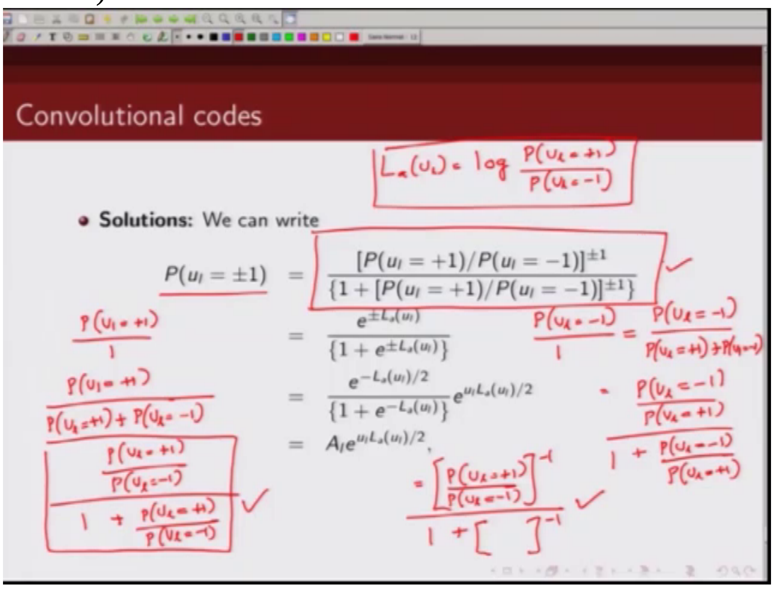
I can write by combining this and this, I will get this. Now if I write these ratios of probabilities in terms of L values, so what is this L value of u l, this is log of probability of u l being plus 1 by probability of u l being minus 1. So this can be then written as

(Refer Slide Time 32:50)



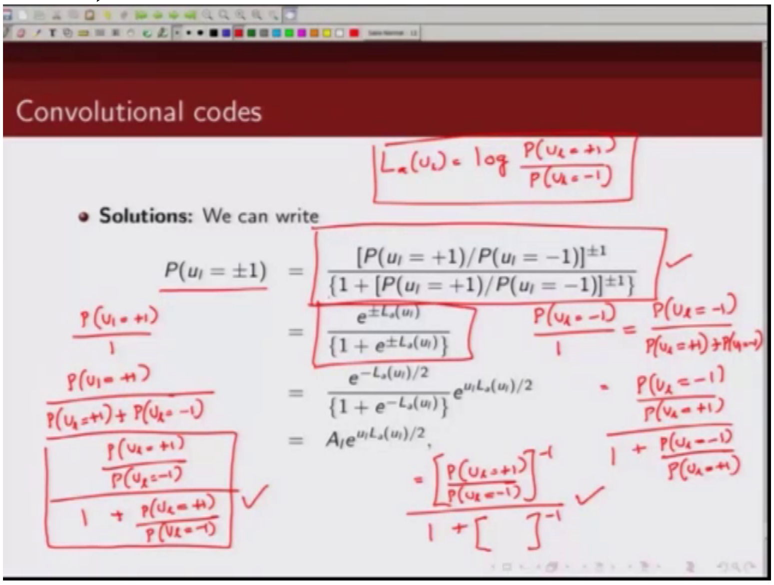
e raised to power L a u l. If I do that, if I plug this in

(Refer Slide Time 33:01)



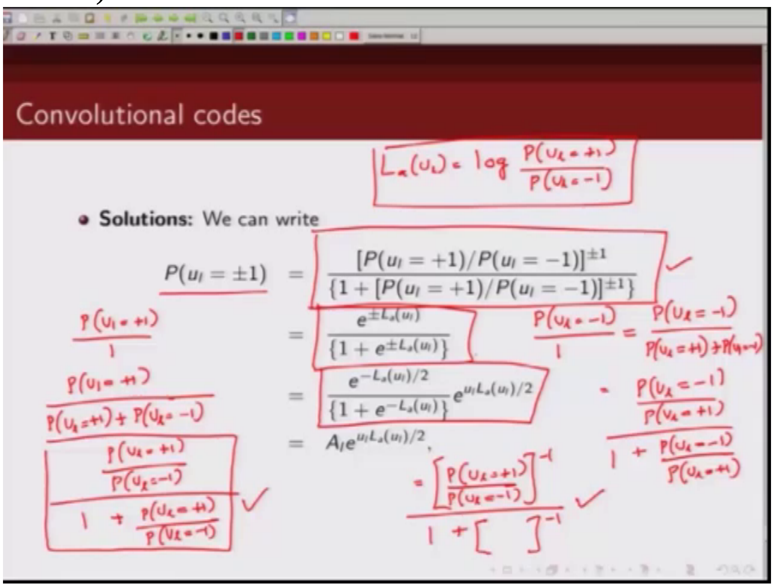
first line what I get here is this term, Ok.

(Refer Slide Time 33:06)



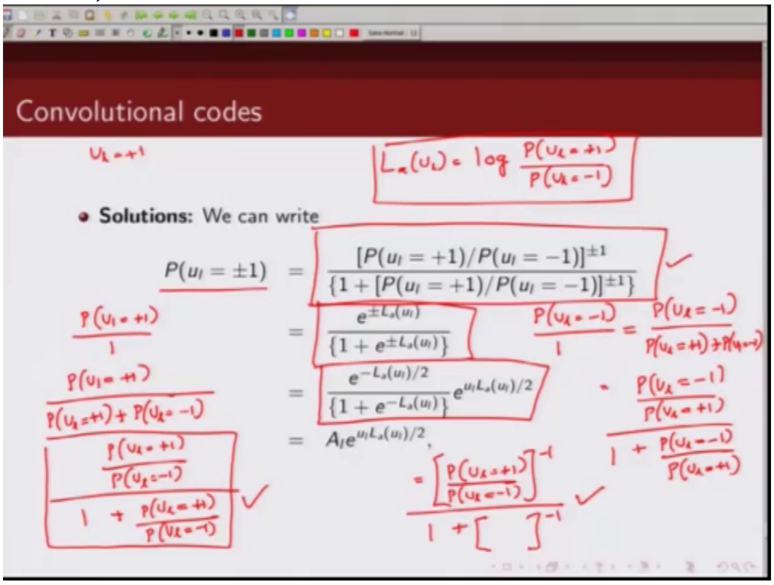
Now note I can further simplify this into this expression.

(Refer Slide Time 33:16)



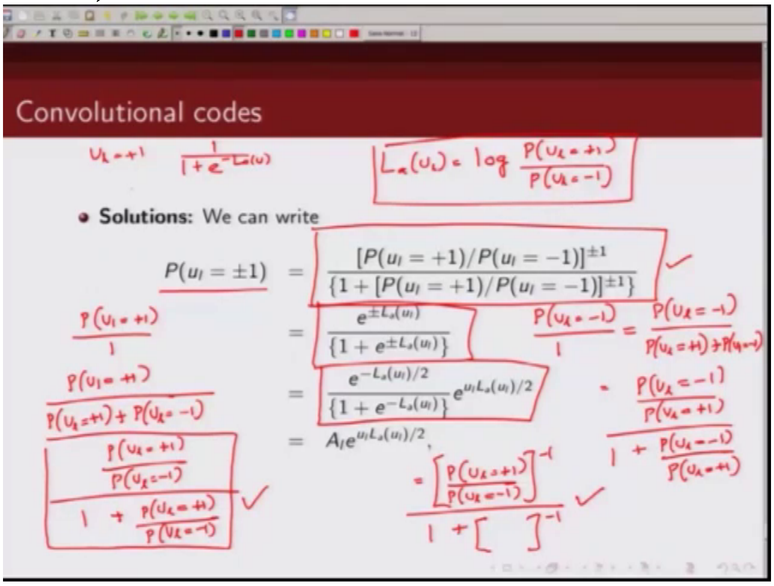
You can see when u l is plus 1, when u l is plus 1, what do we get? When u l is plus 1,

(Refer Slide Time 33:26)



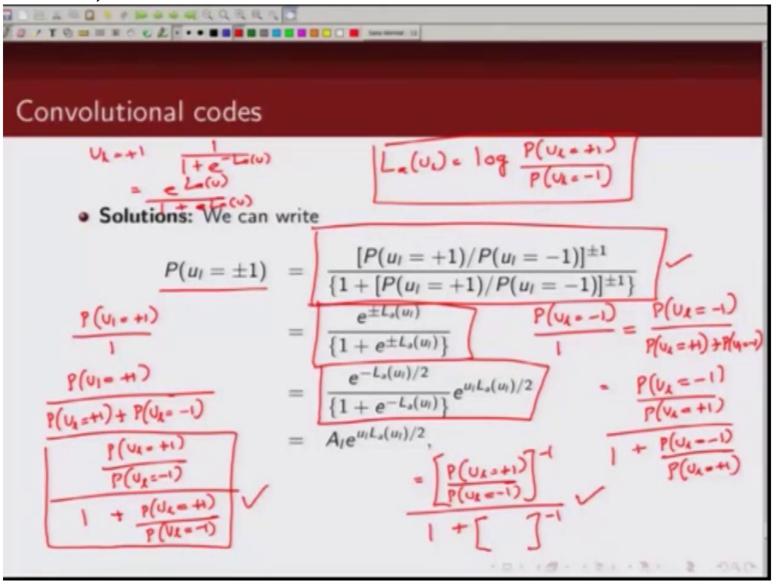
this is e raised to power L a u l by 2 and e raised to power minus L a by 2. So this will be basically 1 so this will be 1 times 1 plus e raised to power minus L a u which can be

(Refer Slide Time 33:44)



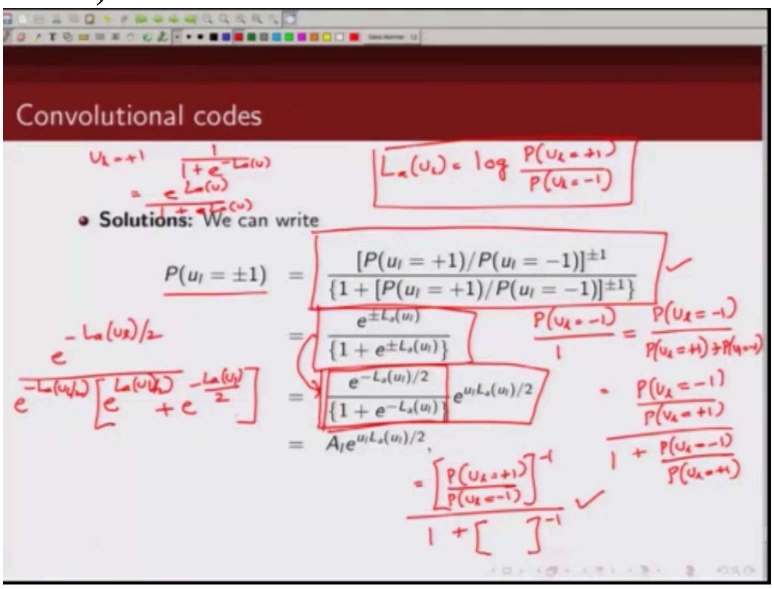
written as e L a u 1 plus e L a u. This is

(Refer Slide Time 33:52)



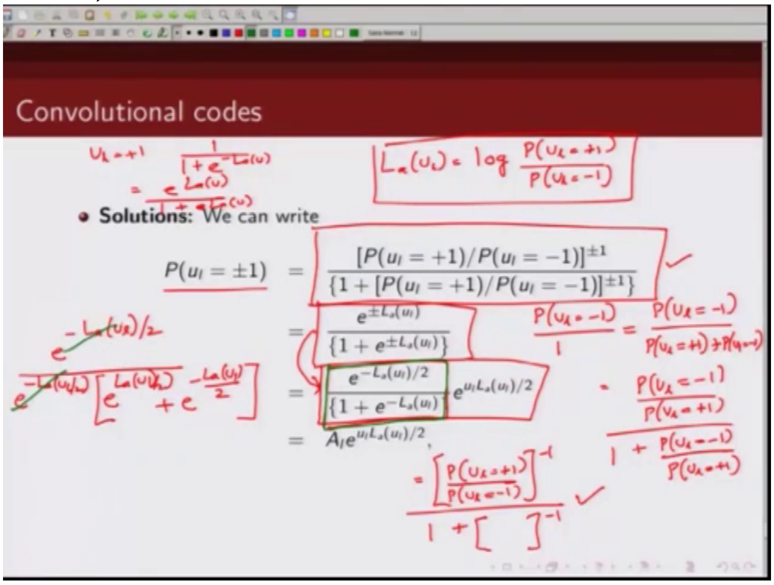
precisely what I have written here. And if u l is minus 1, this will be e raised to power minus L a u by 2 and e raised to power L a u by 2. So this term will become, in that case, e raised to power minus L a u 1 plus e minus La u l. So this term can be written in terms of this, right? And what is this term? What is this term? This I can simplify this term, let's make some space. I can simplify this term as e raised to power minus L a u l by 2 and I have e raised to power minus L a u l by 2, this is e raised to power L a u l by 2 plus e raised to power minus L a u l by 2.

(Refer Slide Time 35:03)



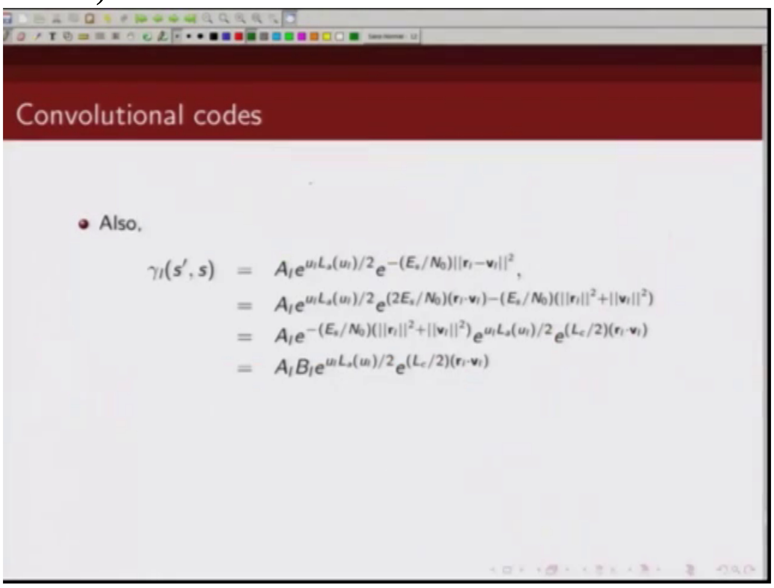
So what I am doing here is I am writing this particular term. So this I can write as this and this. So this cancels out.

(Refer Slide Time 35:18)



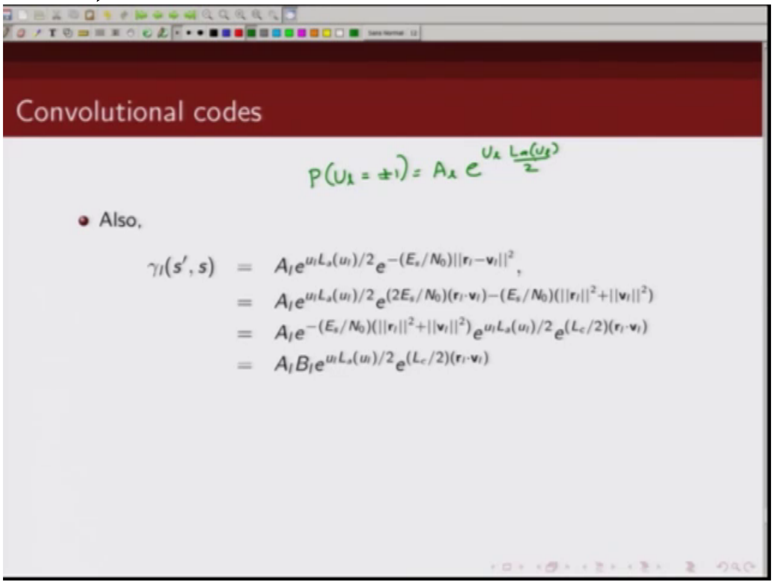
And this is e raised to power x plus e raised to power minus x. This will be cosh of x and that's the symmetric function, so that does not depend on sign of u l whether u l is plus 1 or minus 1, it does not depend on that. I can write this in terms of this expression.

(Refer Slide Time 35:41)



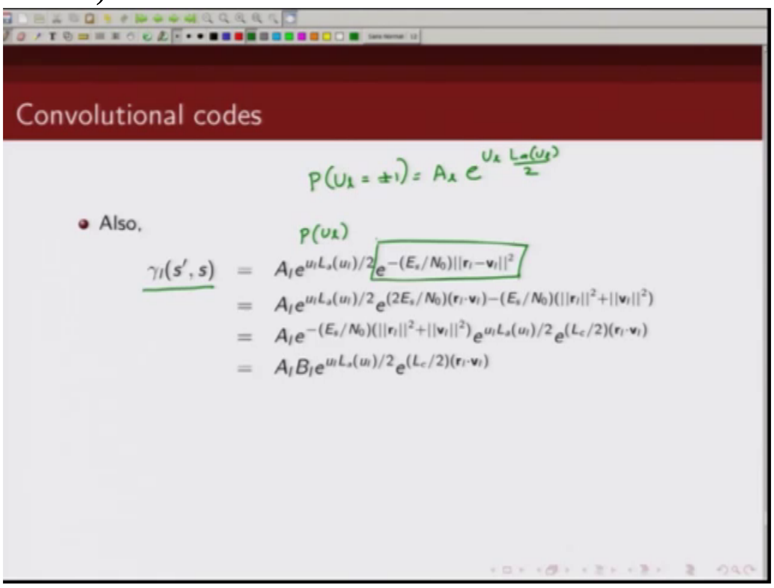
So we will use the expression that we derived in the previous slide for a priori value which was u l being plus 1 or minus 1 as a L e raised to power u l L a u l by 2. We will use this

(Refer Slide Time 36:03)



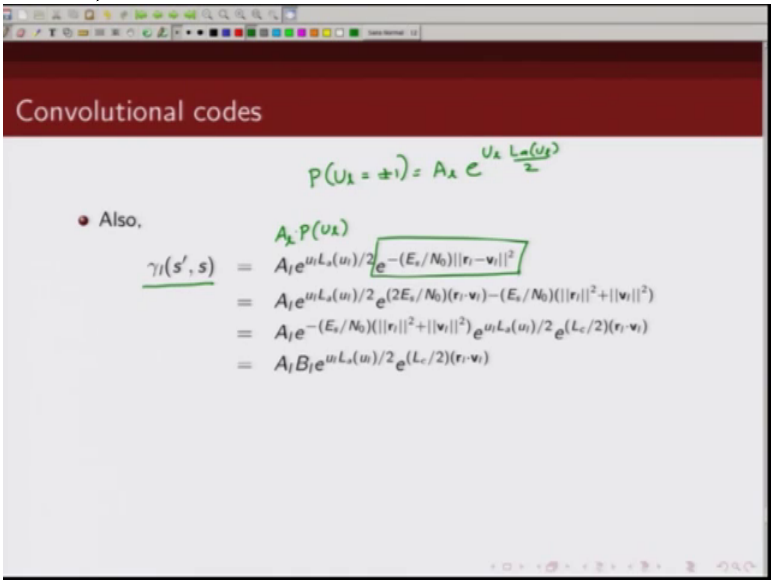
expression to simplify the expression for our branch metric for our b c j r algorithm. Now note if you recall we have written the expression for branch metrics as a priori probability u l and then we had, for a w g n channel we had this expression and

(Refer Slide Time 36:30)



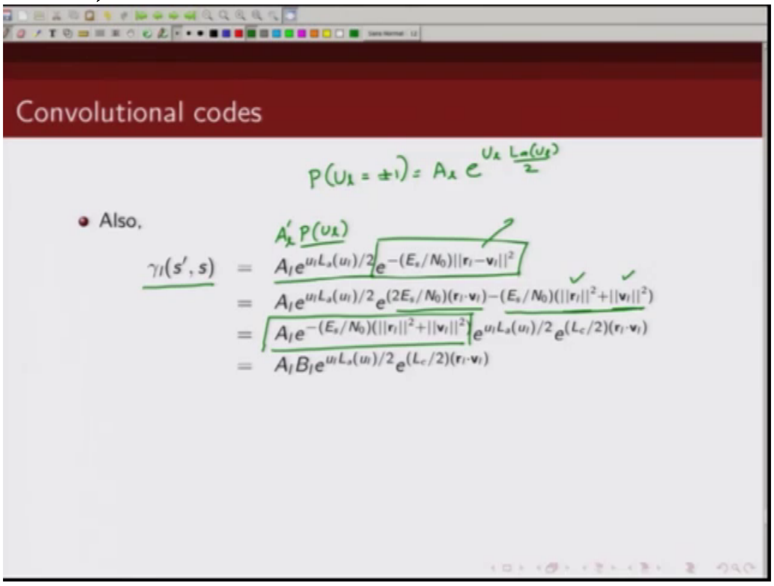
of course there was some constant factor which did not depend on u of l,

(Refer Slide Time 36:37)



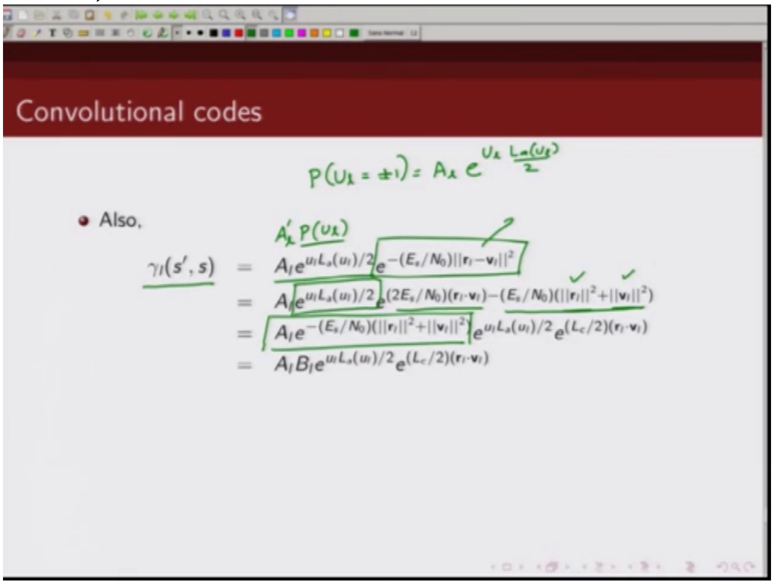
right. So what we did just now was we derived that this a priori probability can be written in this particular fashion, right. Let's further simplify the expression for branch metric. So this we can expand as r square plus v l square plus 2 times dot product or r and v l. Now this does not depend on choice of v of l. And if v of l is mapped to plus 1 and minus 1, v l square will be 1. So this also will be a constant term. So then this term will then not depend on choice of v of l. So what then will we be left with is, so this term we can just take out as some sort of constant which does not

(Refer Slide Time 37:32)



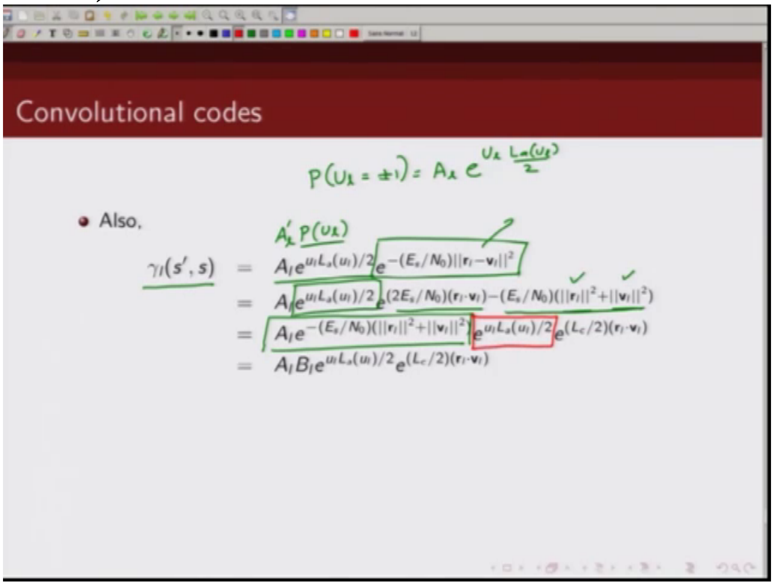
depend on choice of v of l and what will be left is this term which we are

(Refer Slide Time 37:44)



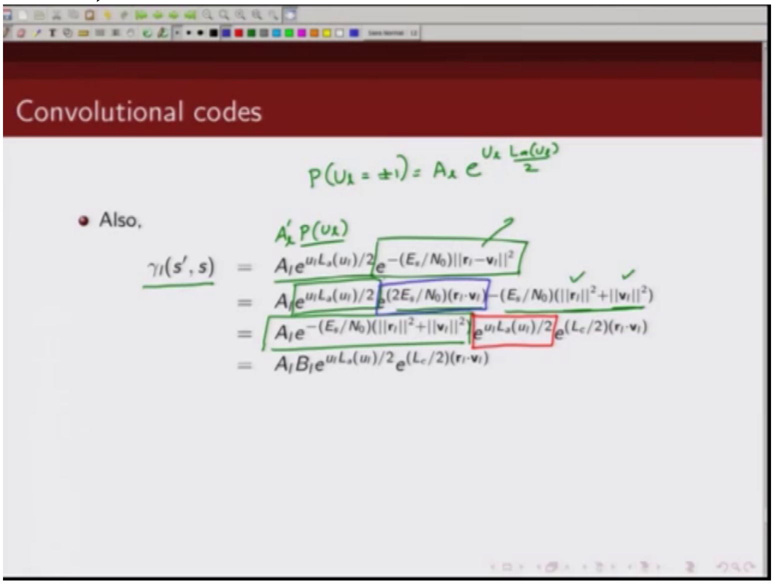
writing here, which we are writing here and

(Refer Slide Time 37:51)



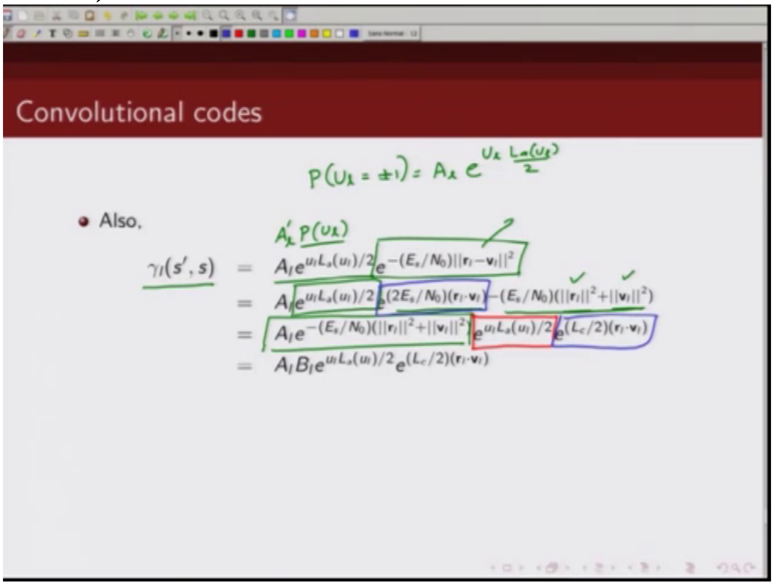
the next term that will be left is this term which we are writing

(Refer Slide Time 37:58)



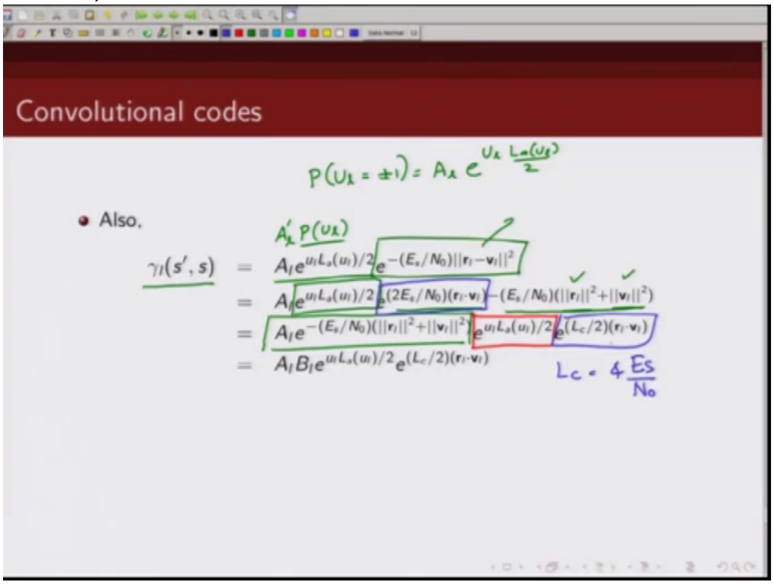
here. Please note

(Refer Slide Time 38:02)



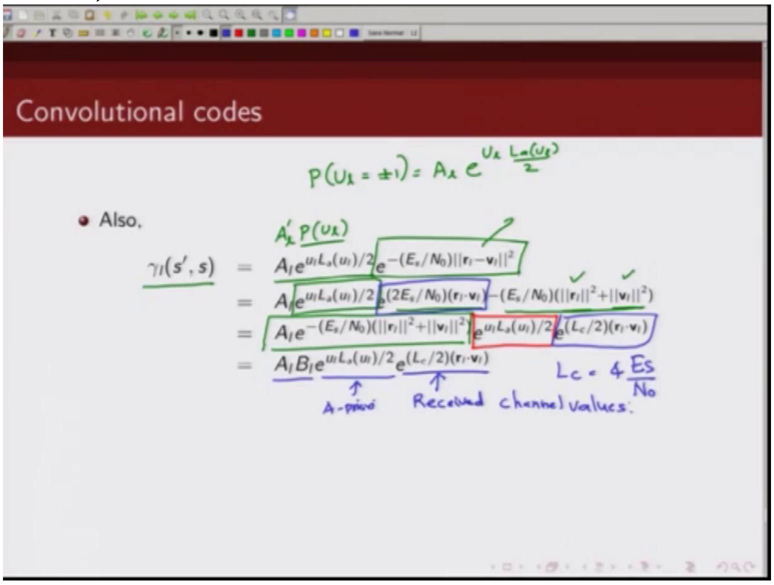
L c is 4 times E s by N naught. So that's why we are writing it as E raised to power L c by 2 and dot product

(Refer Slide Time 38:11)



between r and v l. So then we can just simplify this expression as some constant terms multiplied by this a priori, this a priori term and this is the term which depends on received channel values.

(Refer Slide Time 38:37)

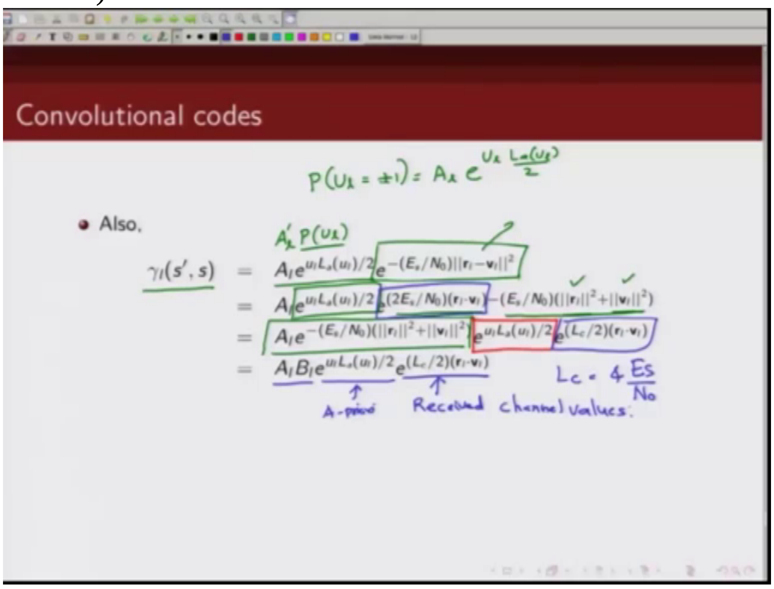


So this is the simplified expression for branch

(Refer Slide Time 38:41)

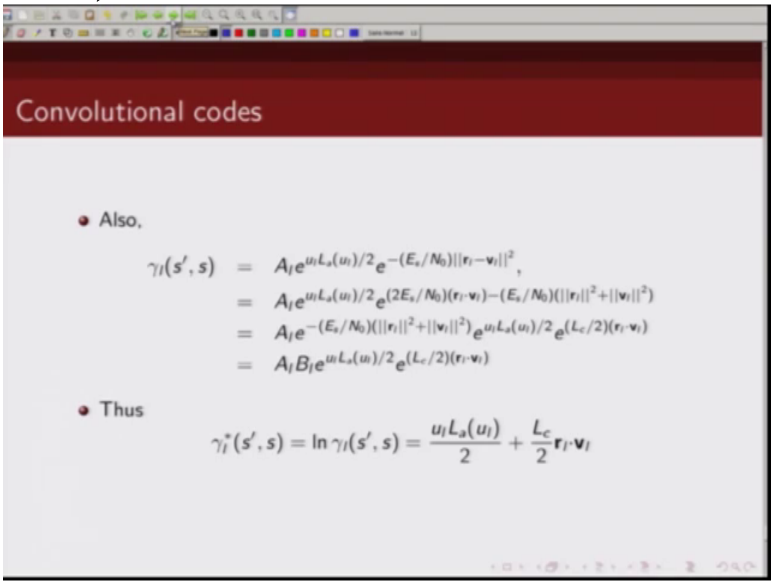
computation for our B C J R

(Refer Slide Time 38:44)



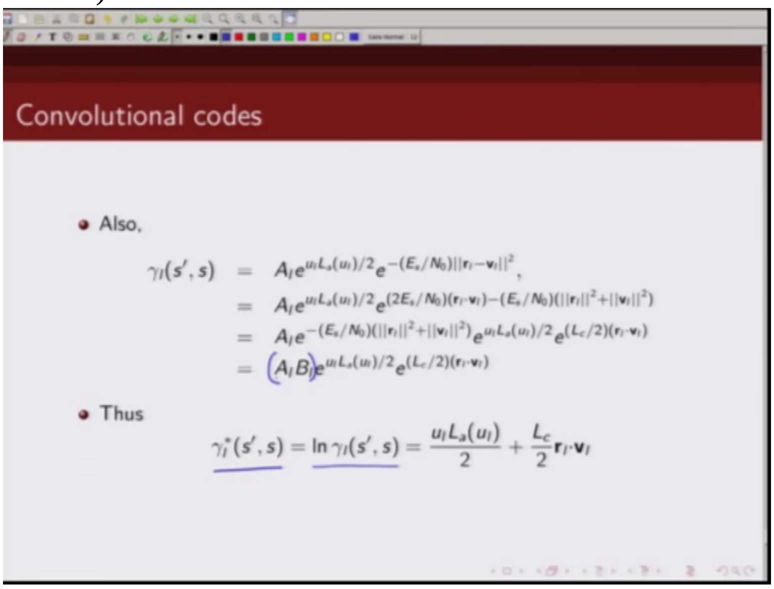
algorithm if we are considering additive white Gaussian noise

(Refer Slide Time 38:49)



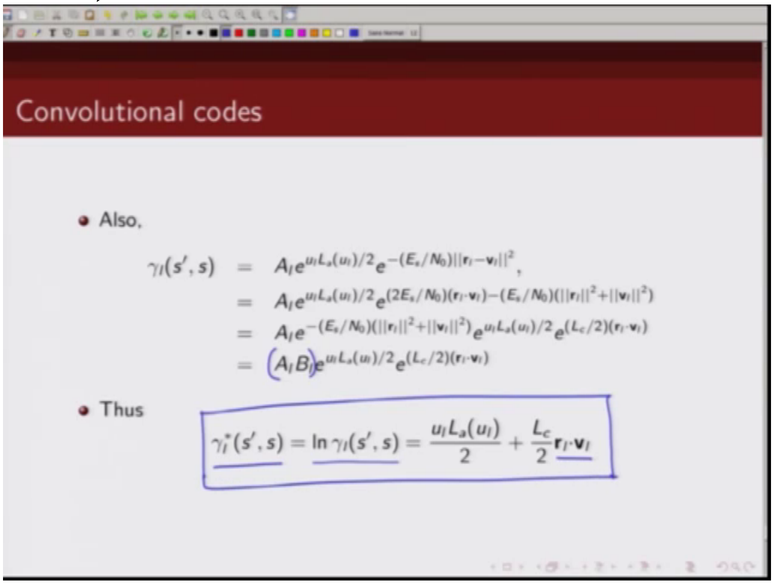
channel. Now if we consider branch metric in the log domain, then log of this term will be some sort of constant, we just

(Refer Slide Time 38:59)



ignore it because this does not depend on choice of v of l, u of l so this will become u l, L value, a priori L value by 2 plus L c by 2 and dot product between the received sequence and the transmitted codeword. So this will be then our simplified expression for

(Refer Slide Time 39:27)



for branch metric computation

(Refer Slide Time 39:30)

for B C J R algorithm over additive white Gaussian noise channel, thank you.