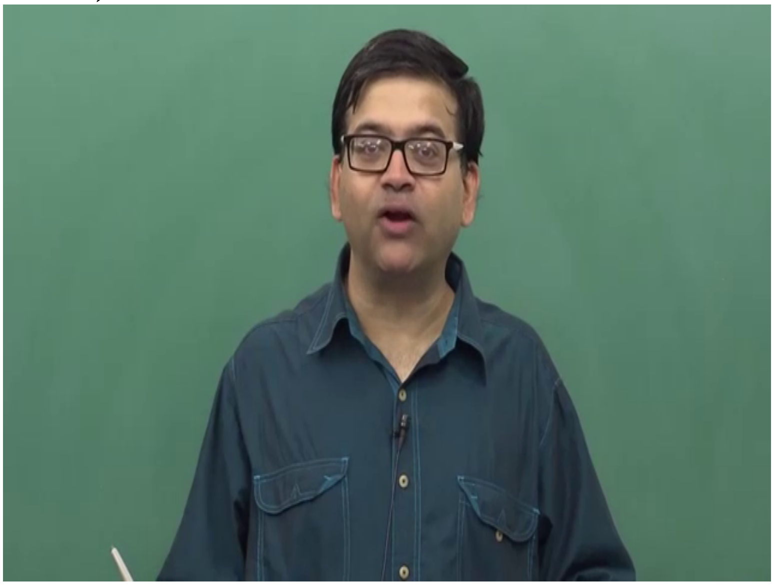
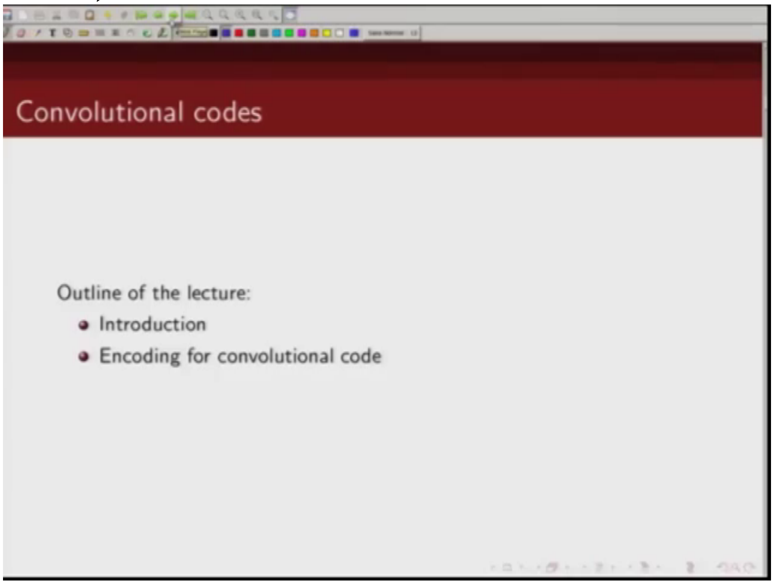
**An Introduction to Coding Theory**
**Professor Adrish Banerji**
**Department of Electrical Engineering**
**Indian Institute of Technology, Kanpur**
**Module 04**
**Lecture Number 15**
**Introduction to convolutional codes-I: Encoding**

(Refer Slide Time 00:18)



So we will start with introduction of convolutional code and today we are going to discuss how we can encode an information sequence using convolutional code. So

(Refer Slide Time 00:26)

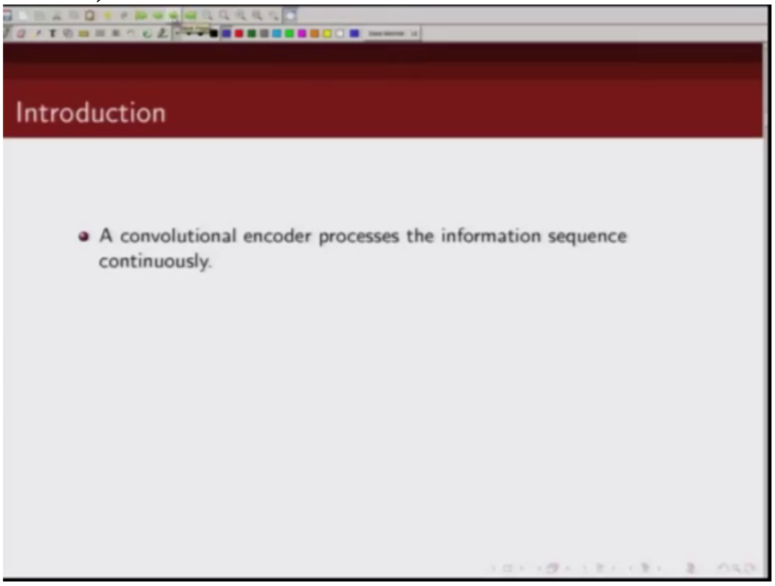



today's topic of discussion is encoding of convolutional code and we will take a very simple

(Refer Slide Time 00:31)

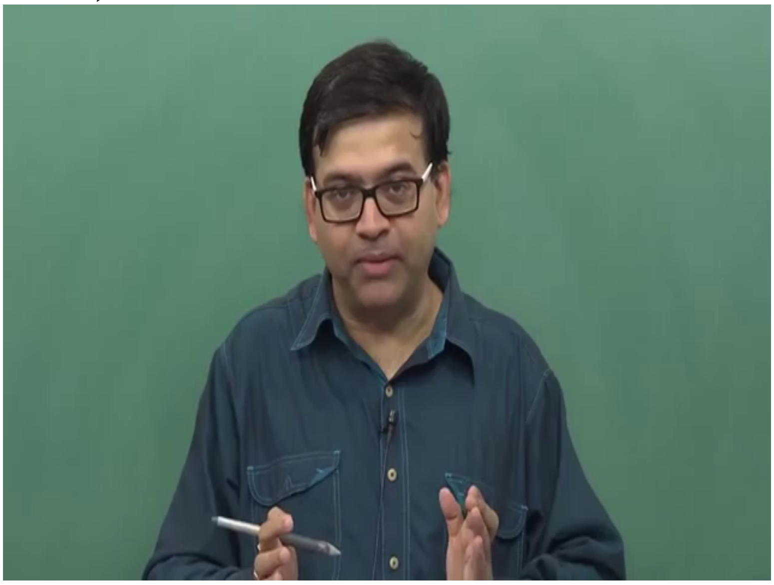
example of a rate 1 by n convolutional code As you know

(Refer Slide Time 00:38)



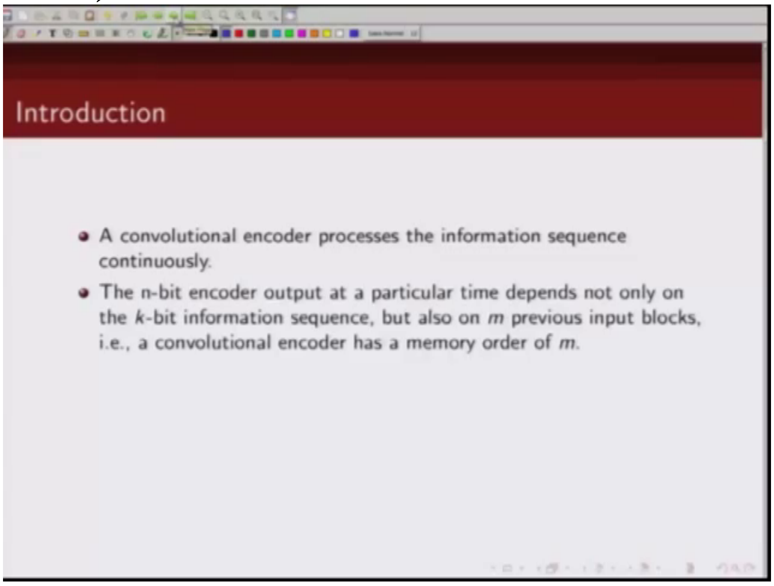

a convolutional code processes information sequence in a continuous fashion. So information bits come in and

(Refer Slide Time 00:46)



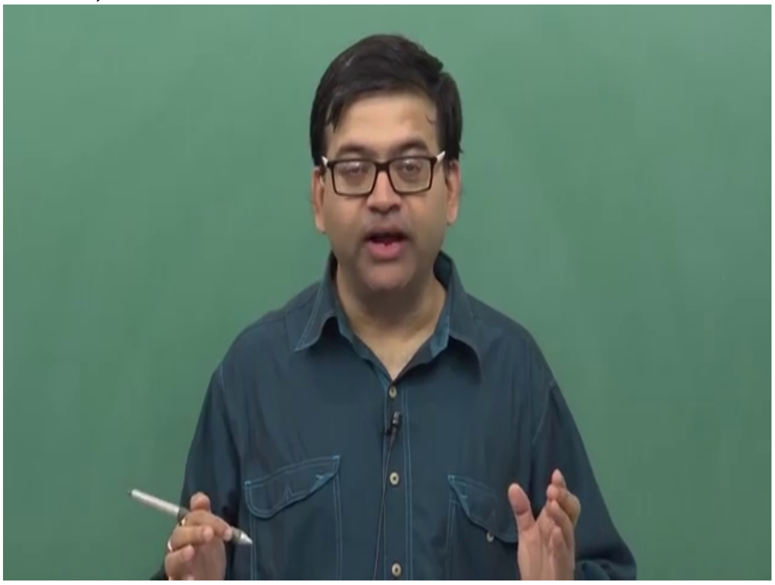
we can get continuous output from a convolutional encoder.

(Refer Slide Time 00:51)



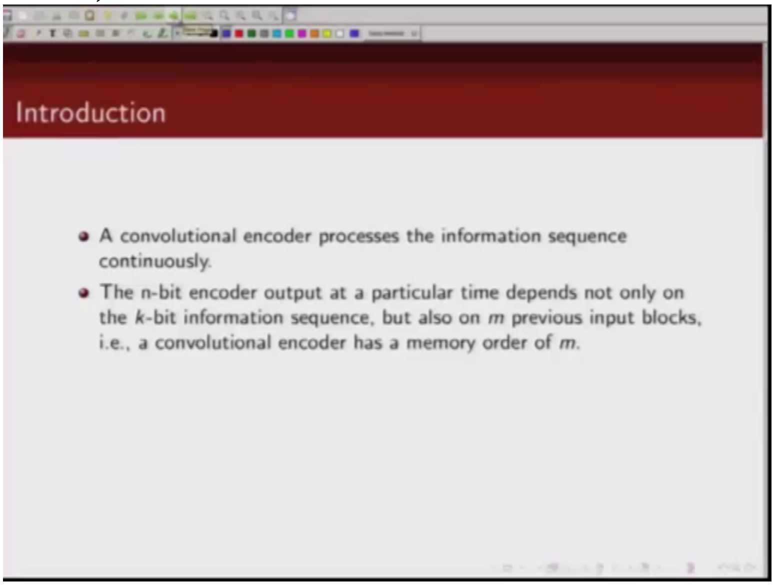

We also know that the output of convolutional encoder depends

(Refer Slide Time 00:58)



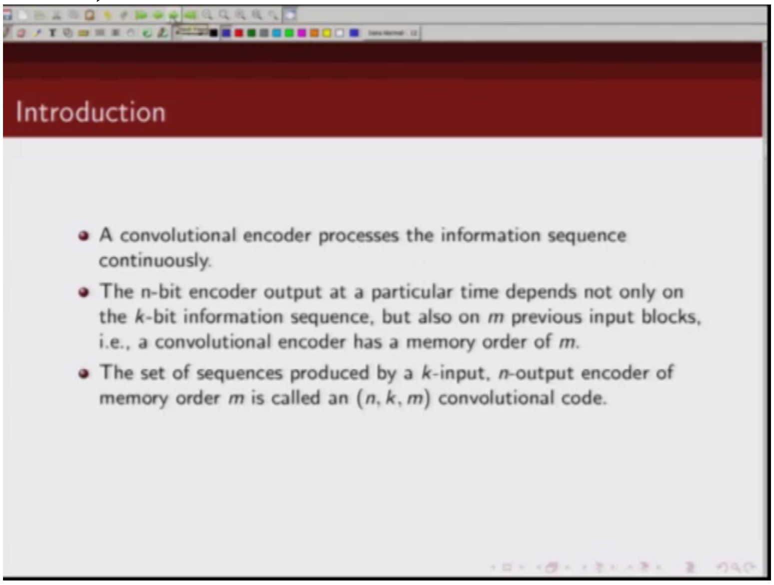
not only on the current input but it also depends on the past inputs and past outputs depending upon the memory of the convolutional encoder. And as we have seen

(Refer Slide Time 01:10)



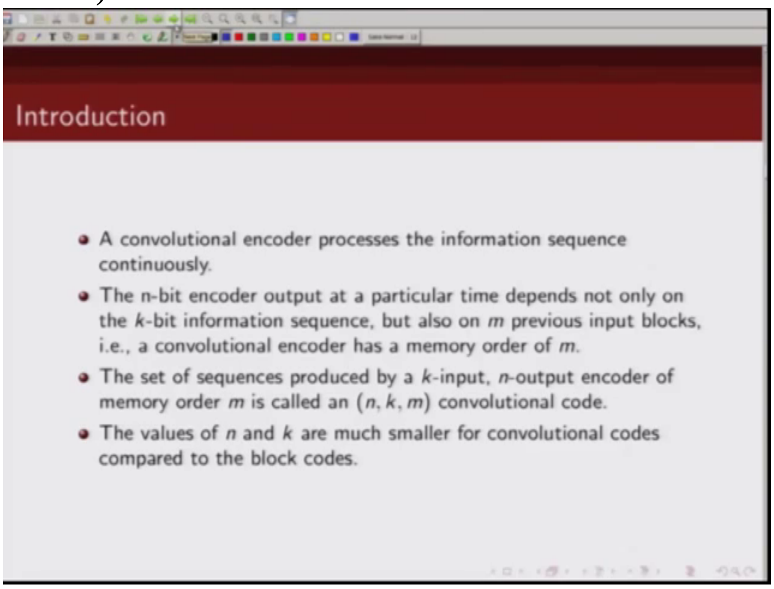

we can realize the convolutional code using shift registers.

(Refer Slide Time 01:16)




So we describe a convolutional code basically as an n k convolutional code with memory m. Now as we

(Refer Slide Time 01:29)

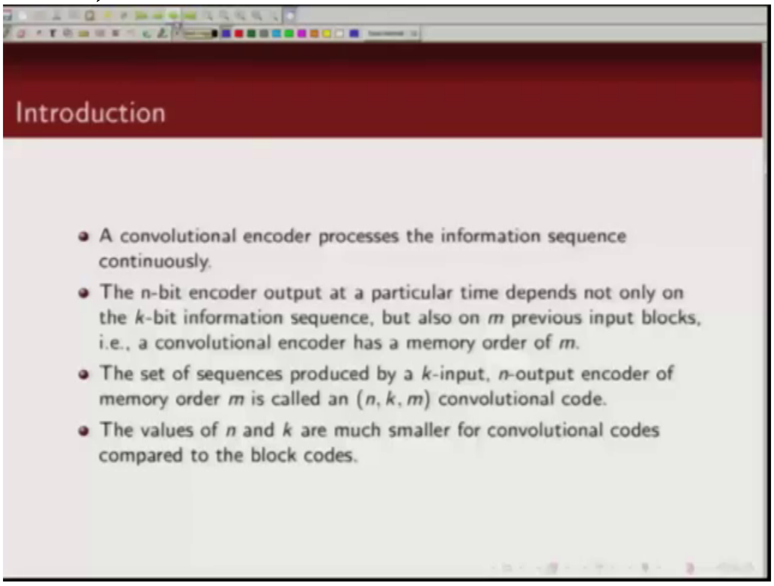



have said before, as opposed to the block codes, typically the value of n

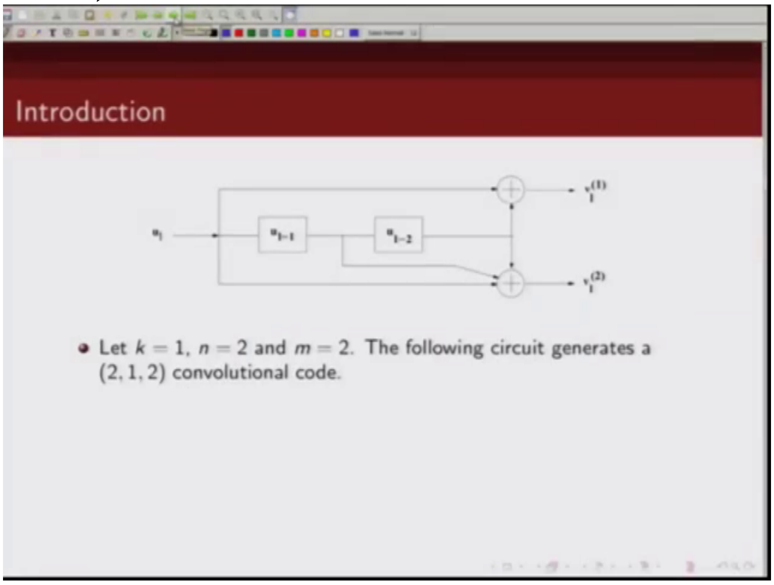
(Refer Slide Time 01:35)

and k for a convolutional code is much smaller, like k may be 1, 2, 3 and similarly n will be, may be 2, 3, 4 like that.

(Refer Slide Time 01:44)

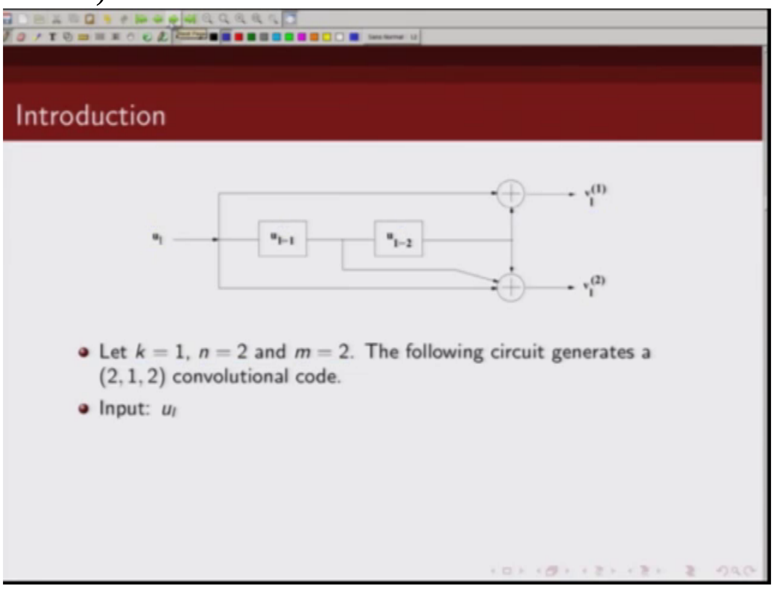


(Refer Slide Time 01:45)



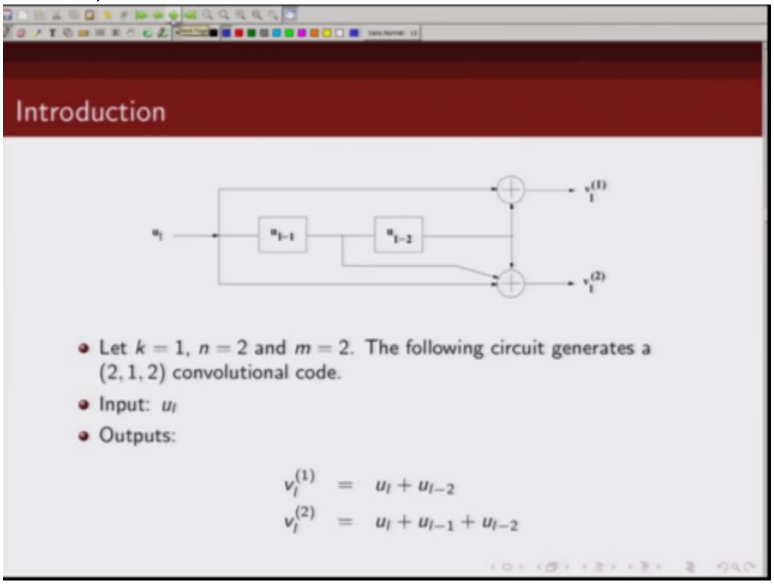

So this is one example of a memory 2 convolutional encoder. We can see this is our input, u of l and output v of l l, and v of l 2. Now note that the output v l 1 and v l 2 depends not only on the current input but also depends on the past inputs as indicated by content here and content here. And this is k equal to 1 because there is only one 1 input here. That's why k is 1. There are 2 outputs this one and this one, so that’s why n is 2 and since the output depends on 2 memory elements, this and this, m is 2. So this is a 2 1 2 convolutional code. So as we

(Refer Slide Time 02:47)



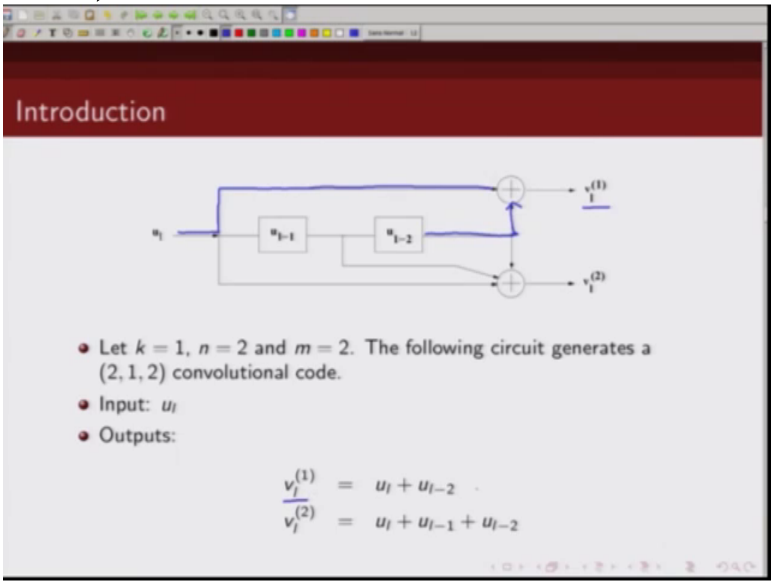

said the input is u of l and output is v of l 1 and

(Refer Slide Time 02:52)



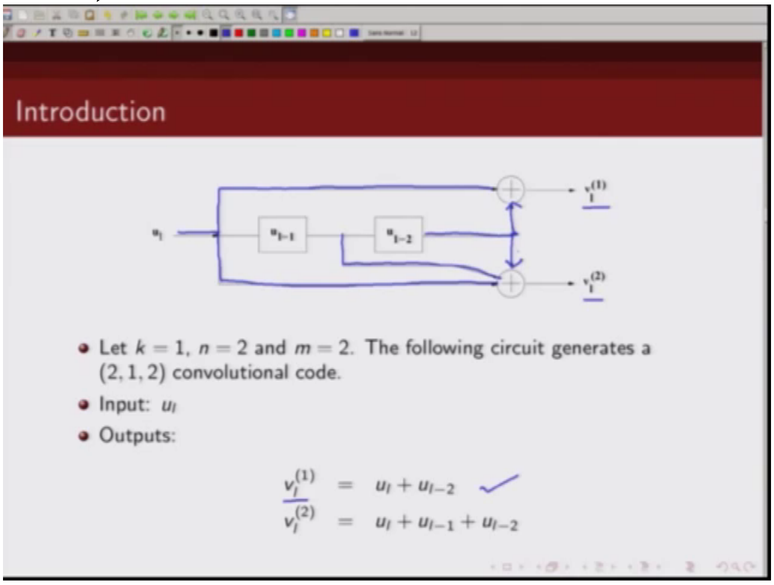

v of l 2 and how is v of l 1 and v of l 2 depend on the u l and the past values? This is given by the interconnection. So we can see for v l 1 it depends on input u l as given by this and it depends on u l minus 2 as given by this link. So v l 1 is given by

(Refer Slide Time 03:17)



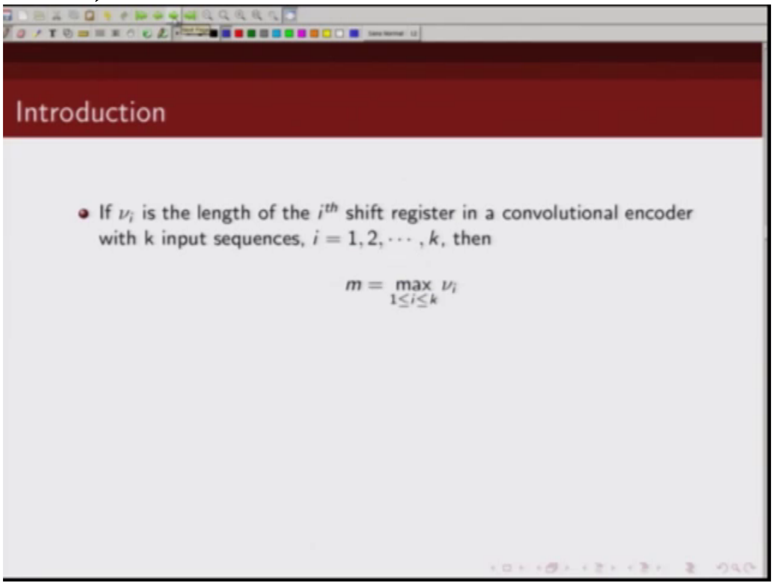

u l and u l minus 2. So it depends on the current input and the input which was there two time instances earlier. Similarly v l 1 2 depends on u l as given by this interconnection, u l minus 1 as given by this interconnection and u l minus 2 as given by this interconnection.

(Refer Slide Time 03:42)



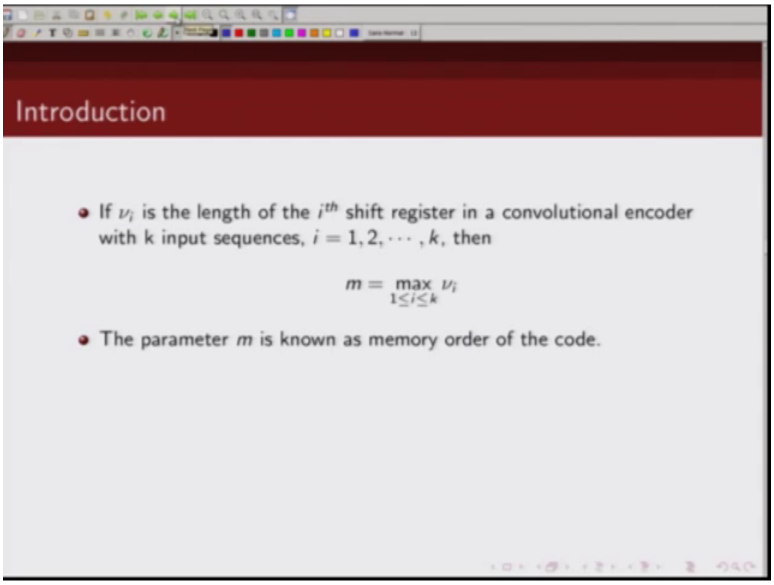

So this is our v l 2. So these are two outputs and this is how they are related to the input. So we can say that whether a particular input appears in the output, that is basically given by these interconnections. These interconnections tell us whether that particular bit is taking part in the output or not. So if

(Refer Slide Time 04:11)



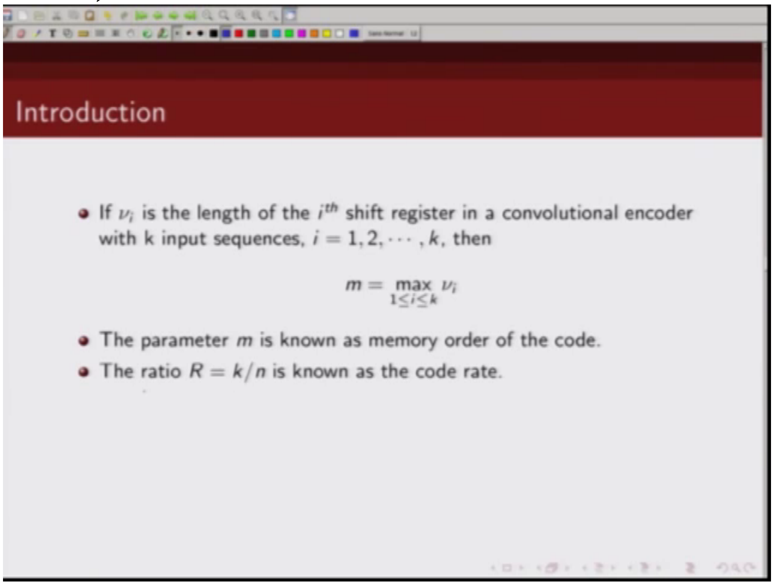

we denote the mu i the length of the i th shift register in the convolutional encoder then we define the memory order as the maximum of, maximum length of the shift register among the k shift registers used to represent the convolutional encoder.

(Refer Slide Time 04:33)



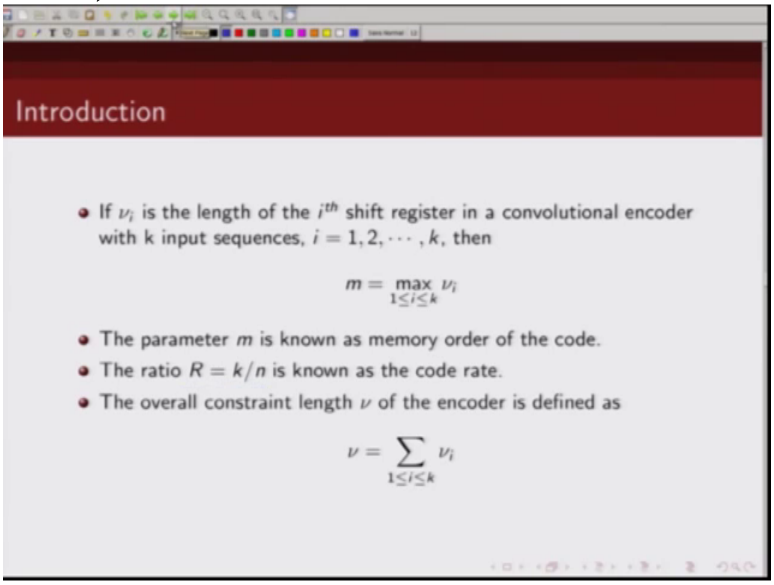

And this parameter m is also known as memory order of the convolutional code. As we know this ratio

(Refer Slide Time 04:45)



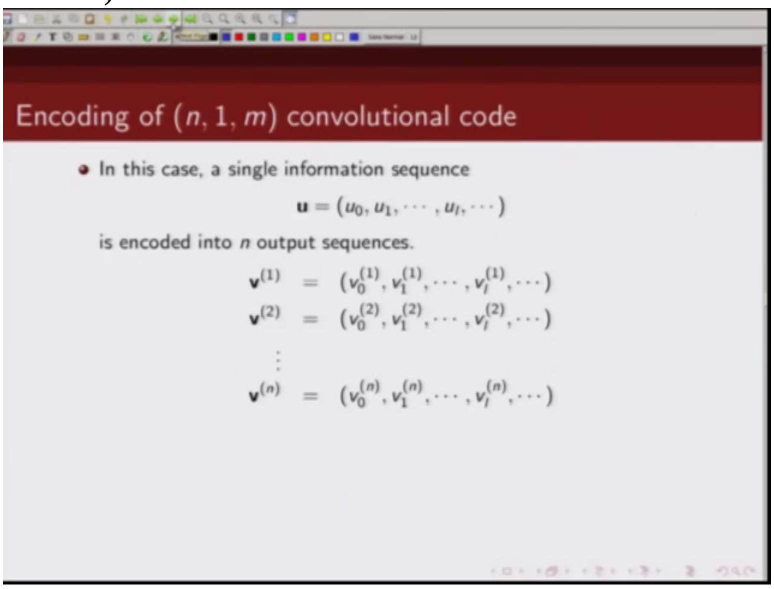

of information bits to coded bits k by n is known as code rate which we denote by capital R,

(Refer Slide Time 04:57)



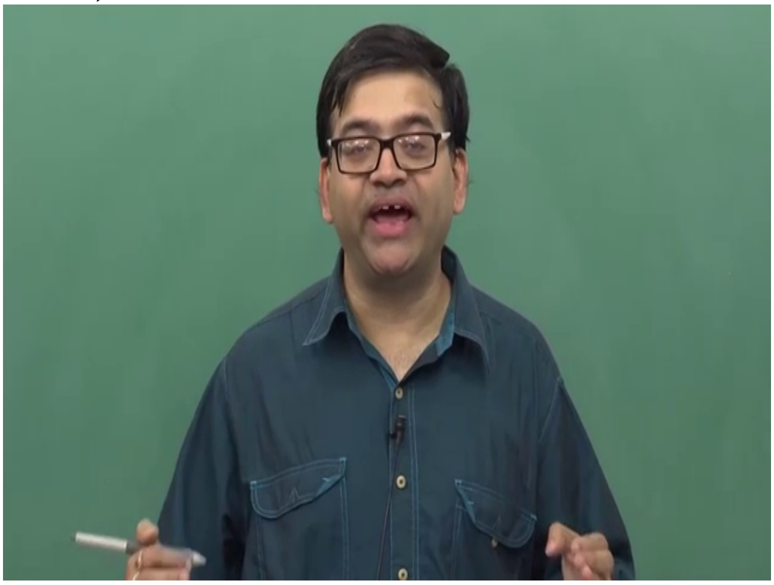

and the overall constraint length is defined as sum of length of all the k shift registers. That's the overall constraint length.

(Refer Slide Time 05:10)



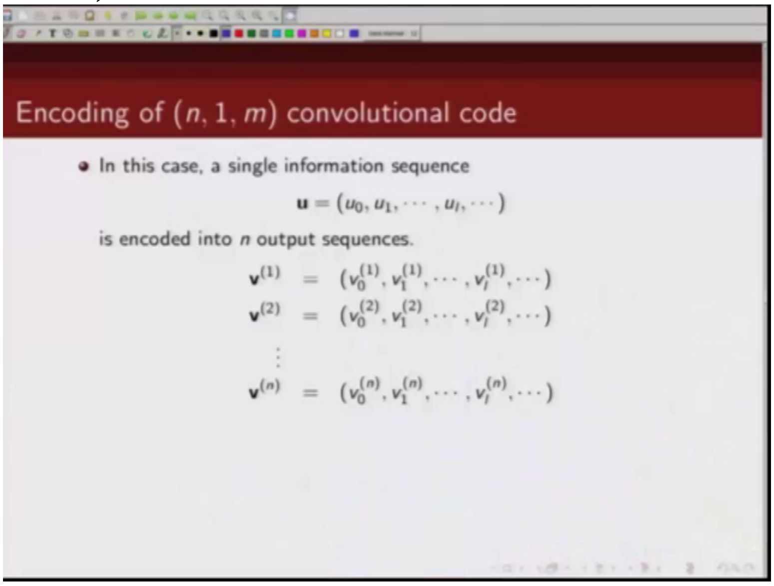

Now we are going to show how we are going to encode

(Refer Slide Time 05:17)



a convolutional or information sequence using a rate 1 by n convolutional encoder. Since it is a rate

(Refer Slide Time 05:25)

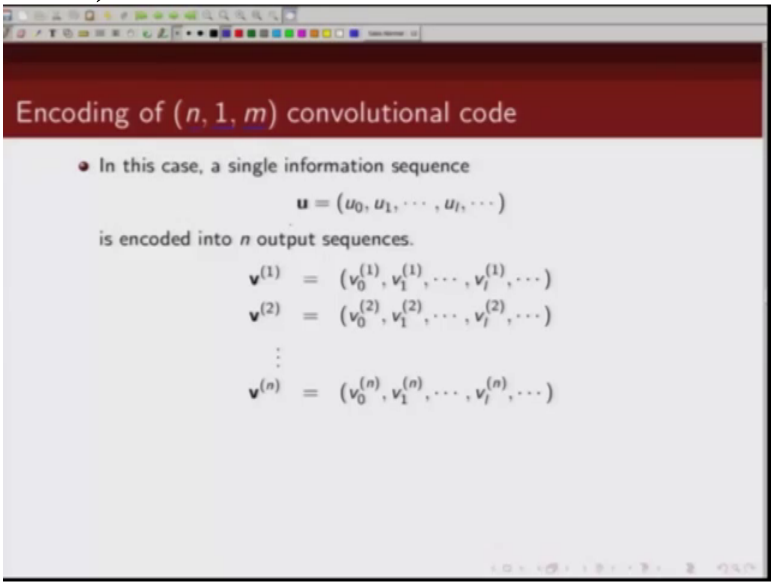



1 by n convolutional code so k is 1 and number of coded bits is n. So there is 1 input coming in and there are n outputs and the maximum length of 1 shift register used to represent this rate 1 by n code is m. So this shift register has m

(Refer Slide Time 05:47)

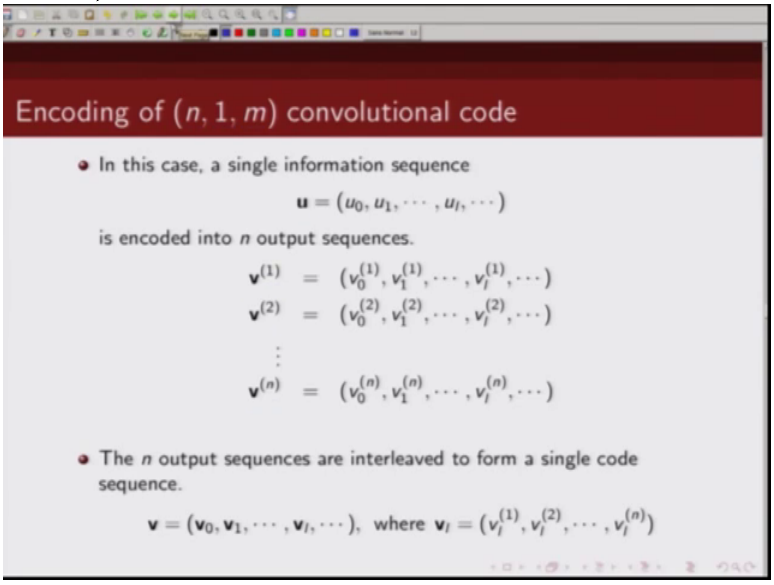
memory elements. So let us take our

(Refer Slide Time 05:51)



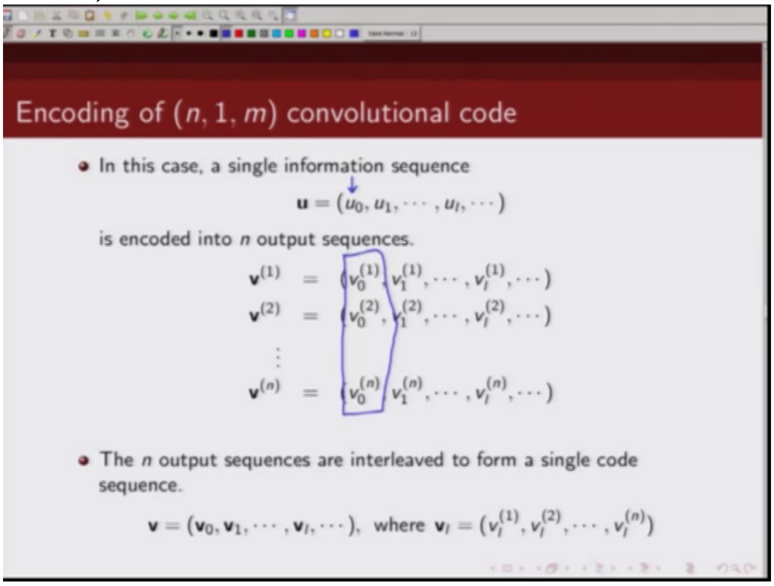

input which we denote by u to be u 0, u 1, u 2, u l minus 1. Since it is a rate 1 by n convolutional code so what we would get is corresponding to 1 input we are going to get n outputs and we denote these n outputs by v 1, v 2, v 3, v n where each of these v i's can be written like this. So the

(Refer Slide Time 06:22)



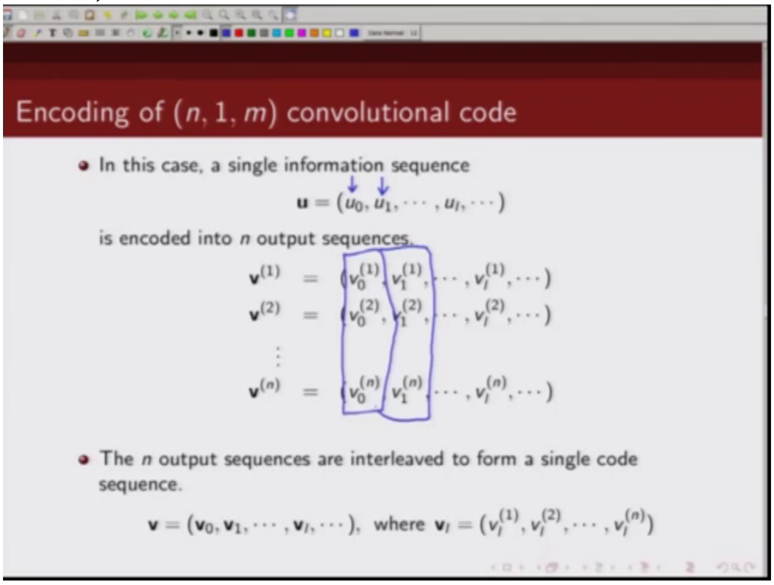

output at a particular instance then is, so corresponding to u 0 then what's the output? These are the n bits

(Refer Slide Time 06:33)



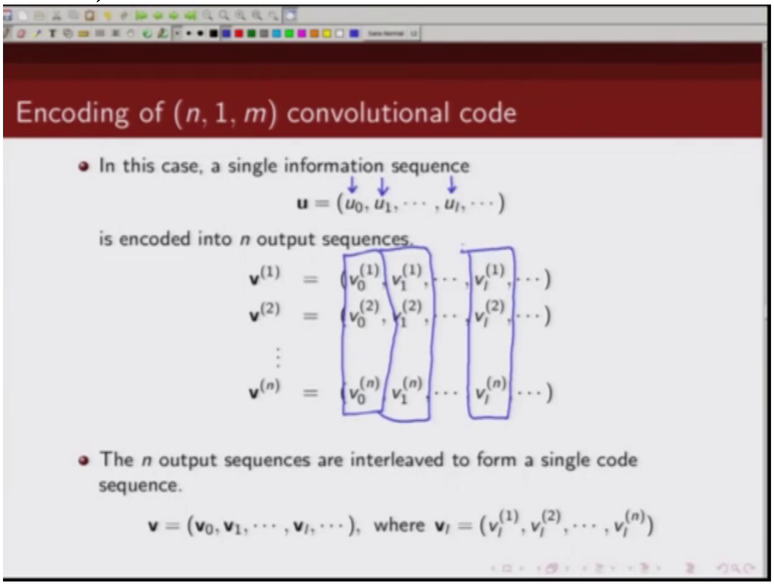

output corresponding to this input u 0. Similarly corresponding to u 1, my output is this.

(Refer Slide Time 06:44)



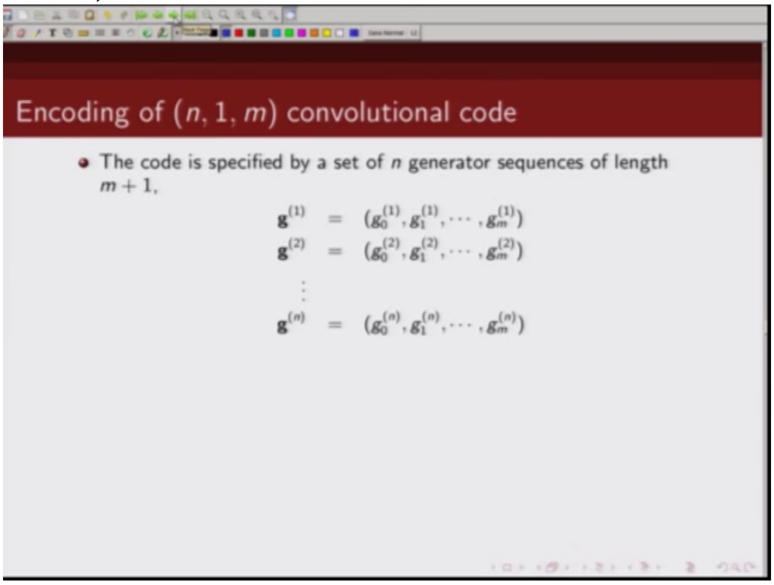

Corresponding to u l my output is this n bit output

(Refer Slide Time 06:50)



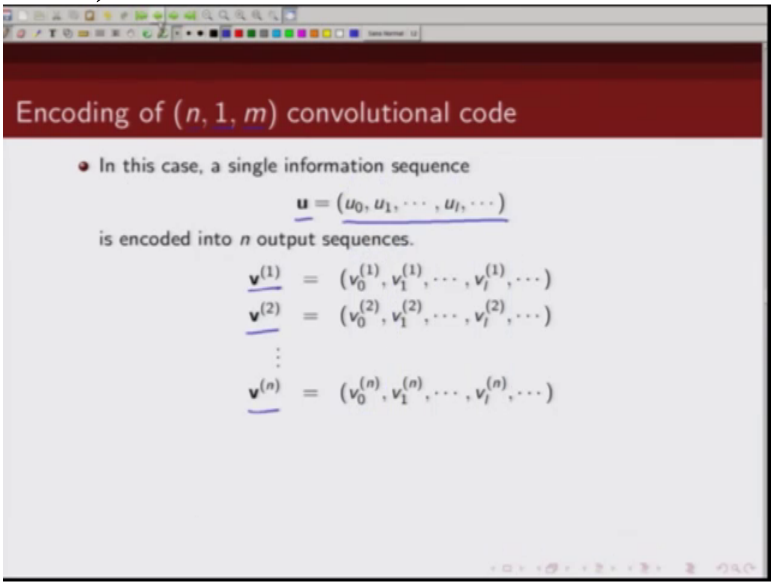

Ok. So I write the output by v 0, v 1, v 2, v l where this v l is an n bit vector. Now

(Refer Slide Time 07:04)



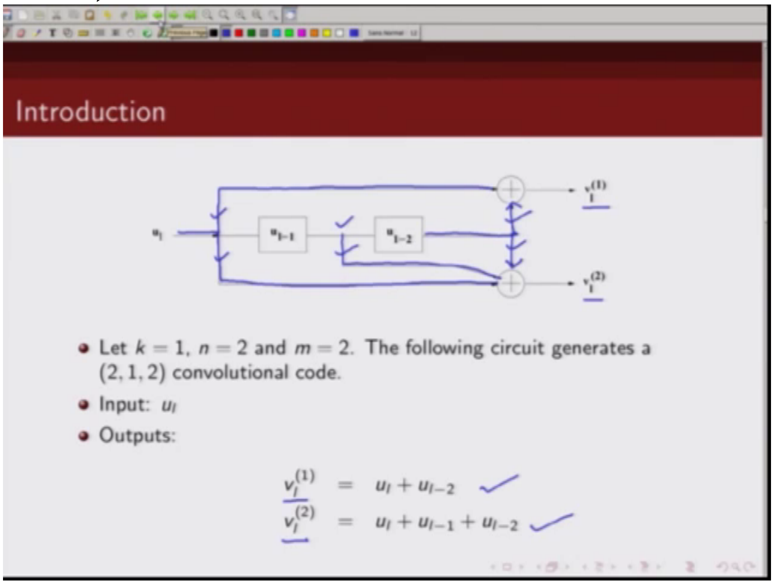

how do we generate these n bit vector from this 1 input and if we just go back to our example

(Refer Slide Time 07:14)



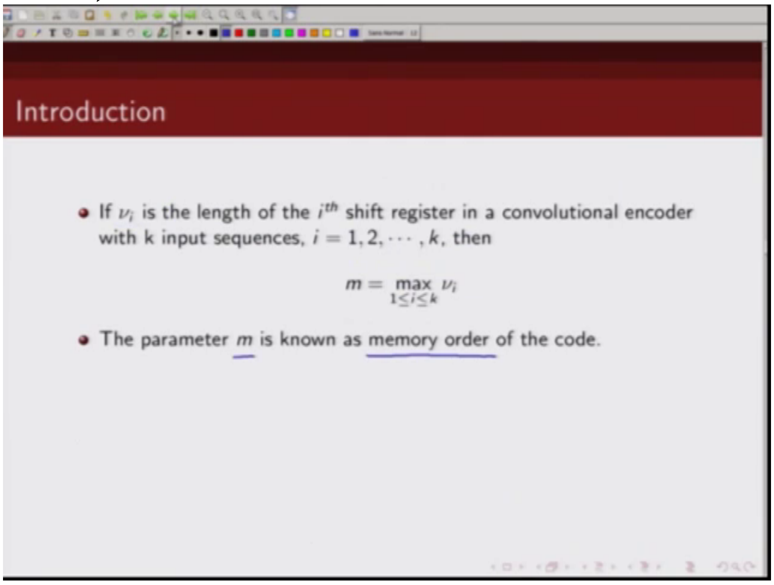

that we had shown, look at this example, how did we generate

(Refer Slide Time 07:20)



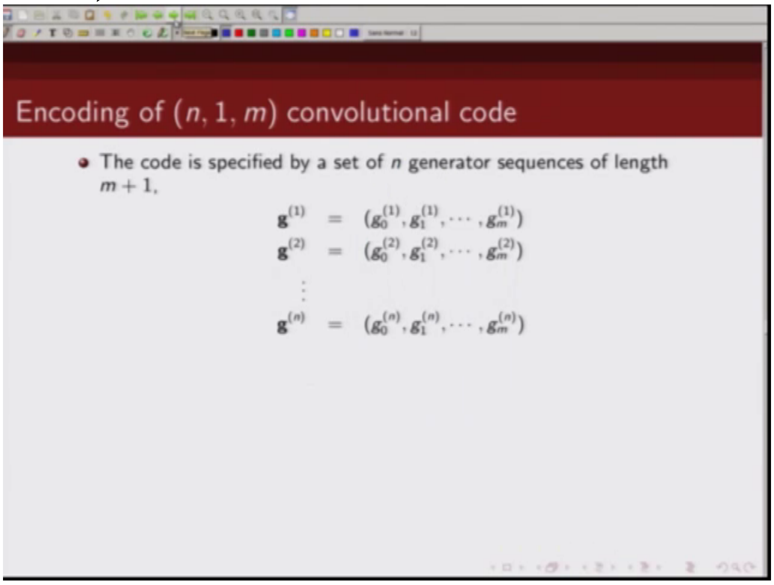

two coded bits corresponding to one information sequence. How did we generate these two coded bits? These coded bits were generated by various combination of input and these past inputs. And whether a particular bit appears in the output that is governed by these interconnections, these interconnections. Whether there is a line connecting this part to the output or not, that determines whether that particular bit, it participating in the output bit. So what we can

(Refer Slide Time 08:02)



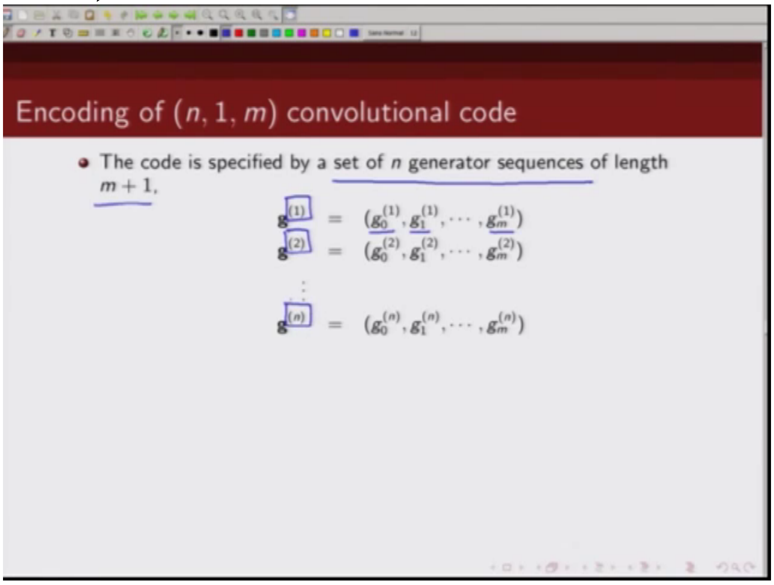

conclude from here is basically we can

(Refer Slide Time 08:07)



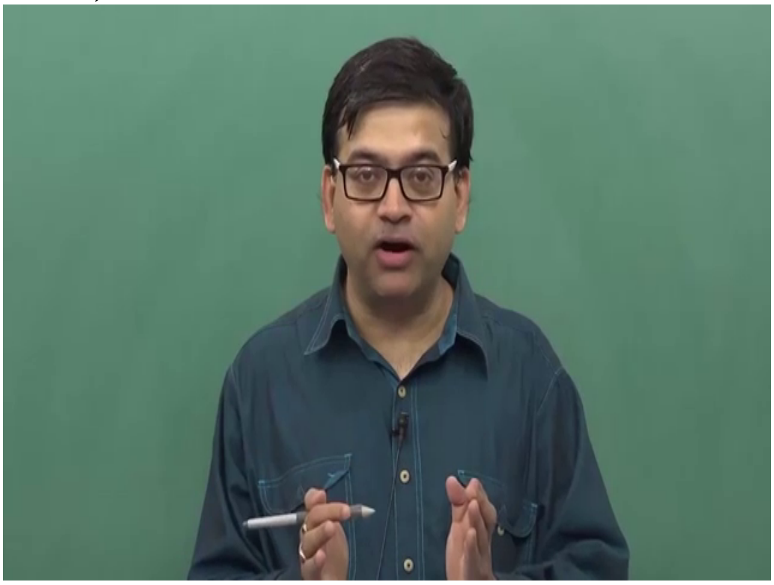

completely specify a code by this set of n generator sequence of length m plus 1 where each of these generated sequence is basically of length m plus 1 and what are these g 0 1, g 1 1, g 1 2, g m 1? So you can see, so this superscript that you see, 1, 2, 3 and n; this corresponds

(Refer Slide Time 08:38)



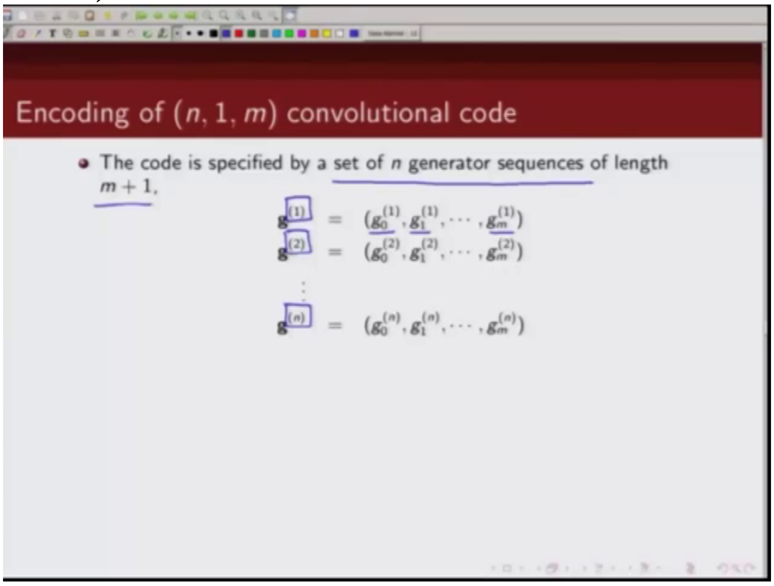

to each of the output sequence. So the first output sequence is specified by this generator sequence g 1. The second output sequence is specified by this generator sequence g 2 and the nth output sequence is specified by this output sequence g n. And what are these g i's. Now note that the memory order of our convolutional encoder

(Refer Slide Time 09:11)



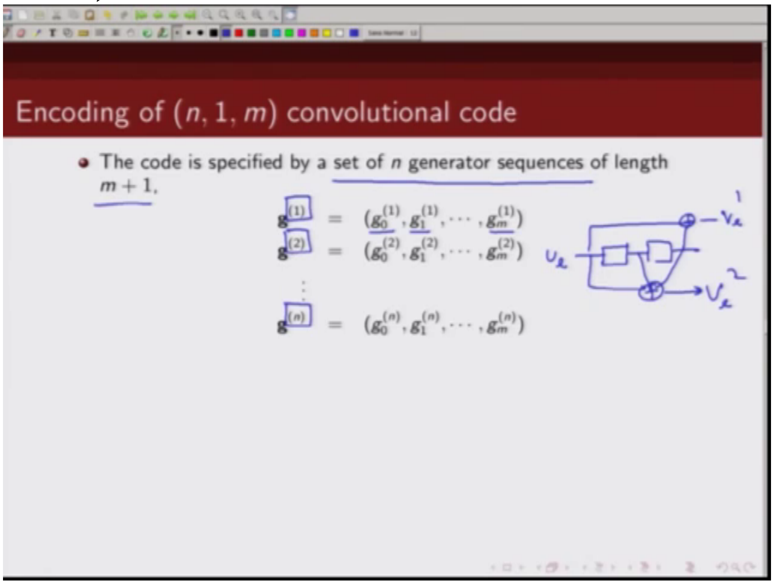
is m. So there are, so if let's say, just take an example

(Refer Slide Time 09:16)



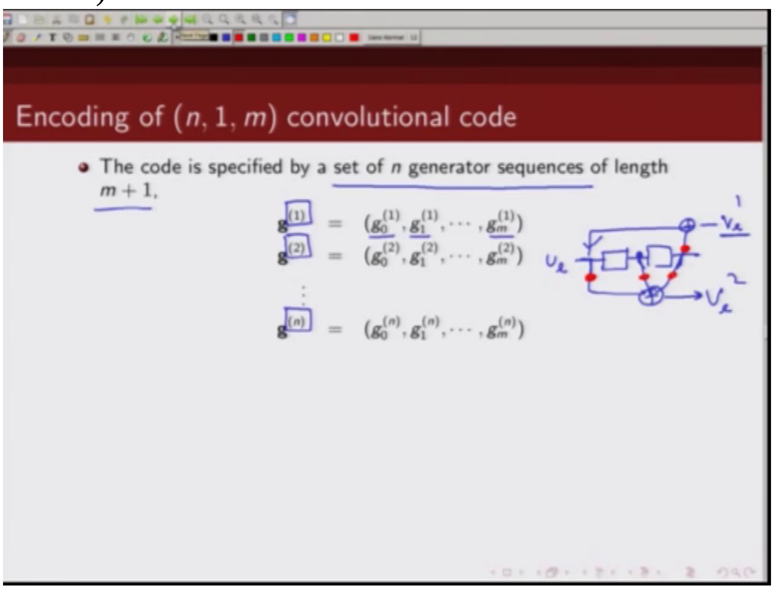

m equal to 2, if we take m equal to 2, let's say 2 memory order, so then basically this is my input u l. And my output I can take from some interconnections from this, let us say this is my example that I had. This was my v l 1; this was my v l 2. Now note

(Refer Slide Time 09:43)



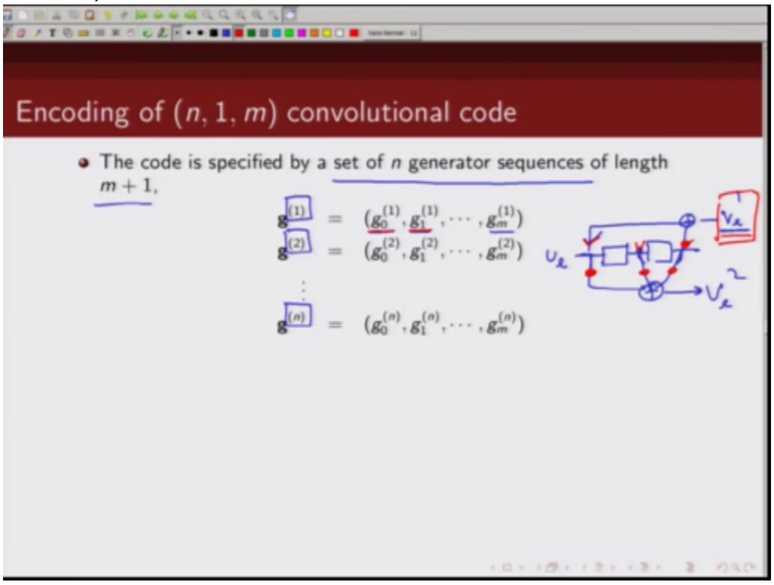

that these interconnections are specifying whether a particular bit is participating in the output code sequence or not. So if we look at the first coded bit u l, now this has memory order m so there are possible m plus 1 connection. What are those possible m plus 1 connection? One, first one is corresponding to whether u l is participating in the output bit or not. Second one is corresponding to whether u l minus 1 is participating or not, this is this point. Third one is this point, whether u l minus 2 is participating or not. Similarly the second coded sequence, whether u l is participating or not, whether u l minus 1 participating or not, whether u l minus 2 is participating or not. So we can see

(Refer Slide Time 10:38)



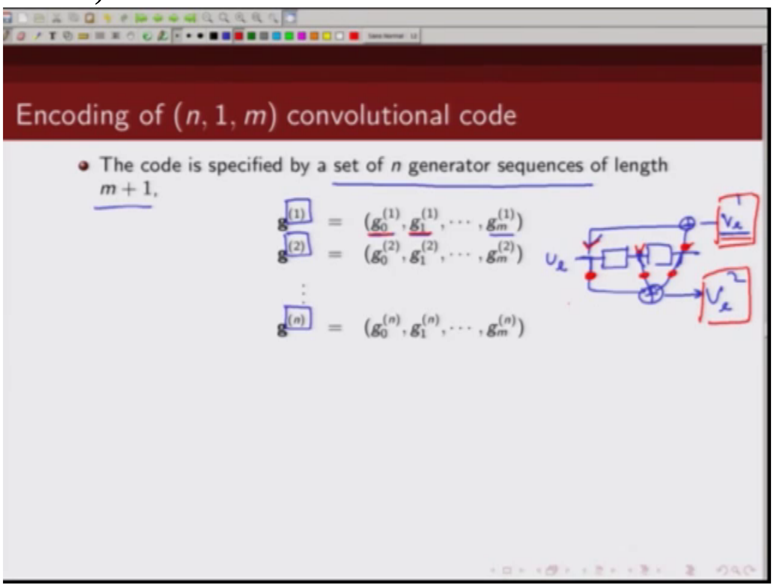

that the output here, let's take the first output sequence, that is completely specified by whether u l is participating, where u l minus 1 is participating, whether u l minus 2 is participating. So in this example v 0 1, v 1 1 and g 2 m, completely specifies what inputs are participating in generating our code sequence.

(Refer Slide Time 11:09)




Similarly look at the second bit. Here also these m

(Refer Slide Time 11:14)

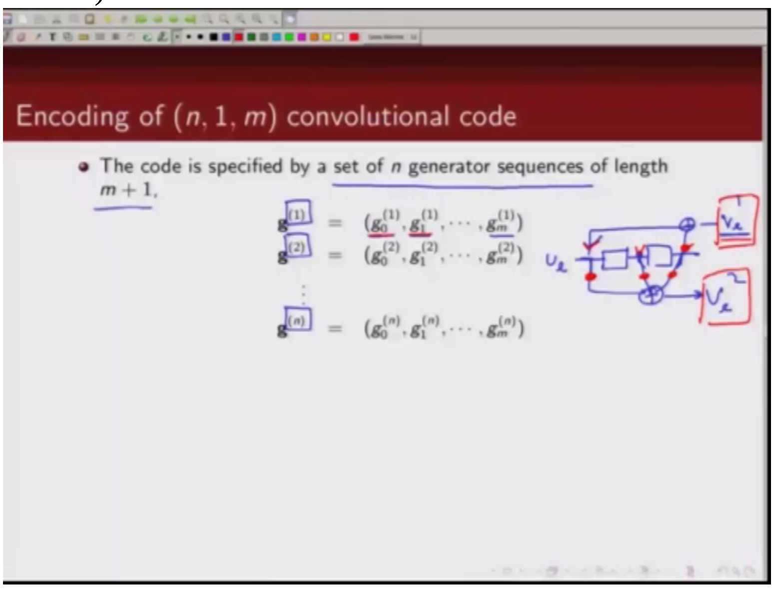



plus 1 connections will completely specify whether the particular bit or the past bits are taking part in the output coded bit. So you can see, if you have a rate 1 by n code whose memory is m then we can completely specify that code using a set of n generator sequence where each of these n generator

(Refer Slide Time 11:43)

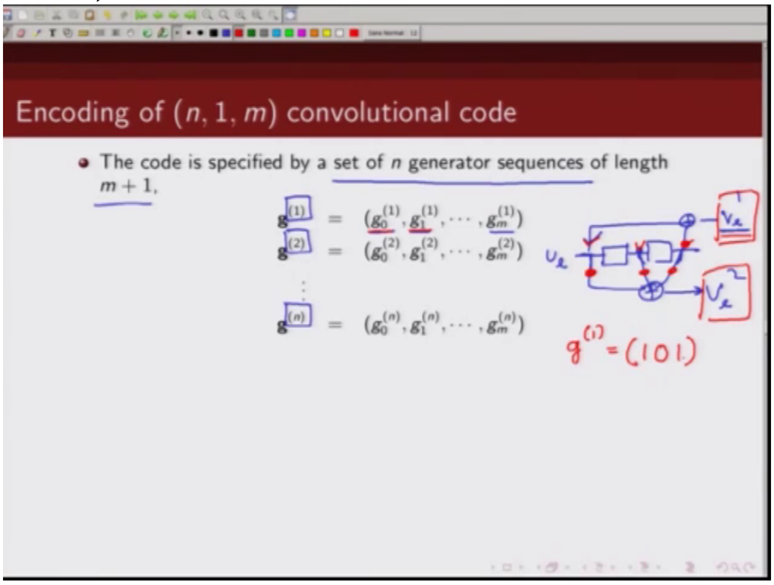
sequence correspond to one of the output sequence and each of the generator sequence

(Refer Slide Time 11:50)



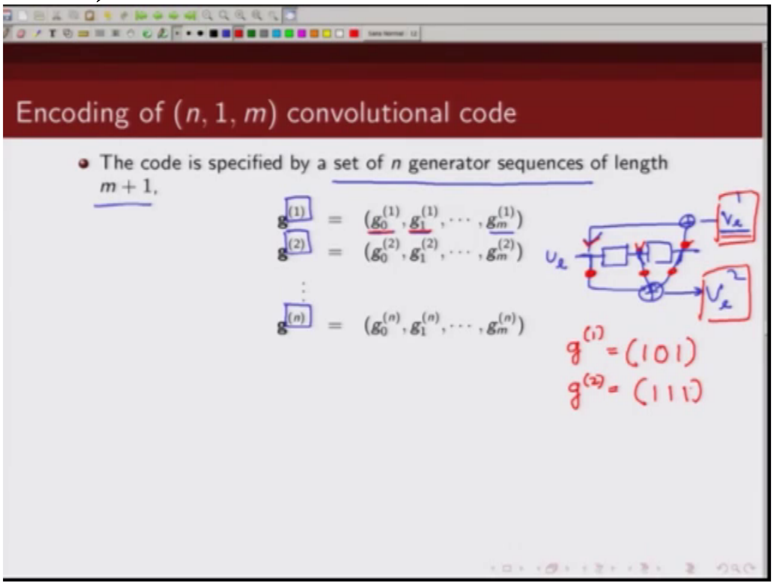

is of length m plus 1 specifying the interconnections of u l, u l minus 1, u l minus 2 up to u l minus m. So then what are these g 0 1 and g 1 1? If g 0 1 and g 1 1 are either 1 or 0, 1 means they are participating, 0 means it does not participate. For example, in this example, what is g 1 1? Is u l participating in the output sequence of v l? Yes it is. So then g 0 1 will be 1. Is u l minus 1 participating in the output sequence v l? No, so then this will be zero. What about u l minus 2? It is participating in the output sequence. So it will be 1. So g 1 is 1 0 1.

(Refer Slide Time 12:47)



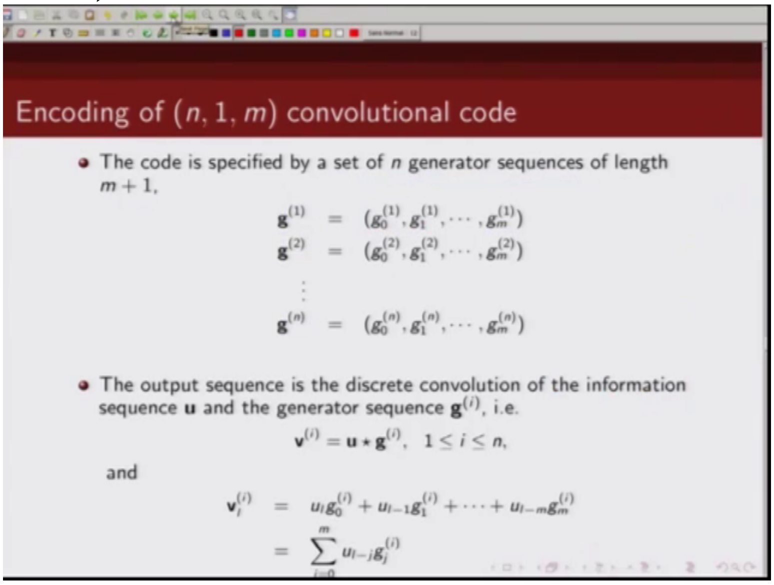

Similarly g 2 will be 1 1 1 because u l,

(Refer Slide Time 12:54)



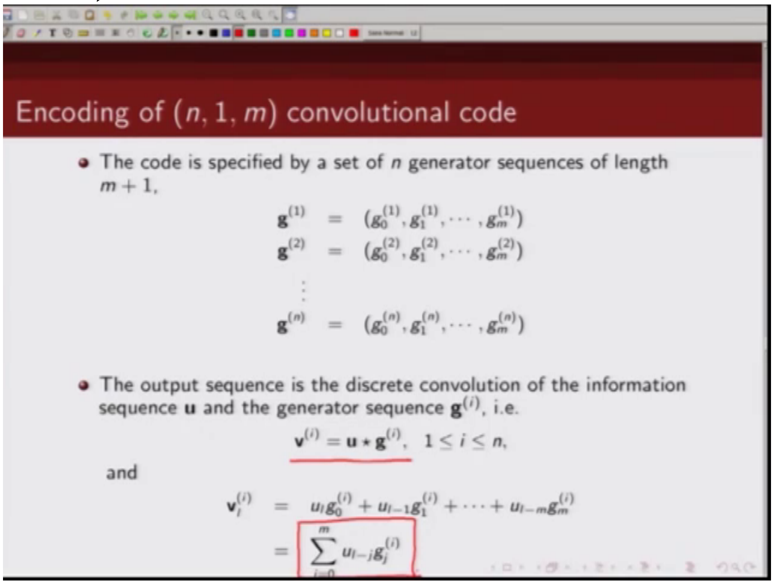

u l minus 1, u l minus 2 they are all participating in the output coded sequence, Ok. So if I specify these generator sequences then my convolutional code is completely specified.

(Refer Slide Time 13:12)



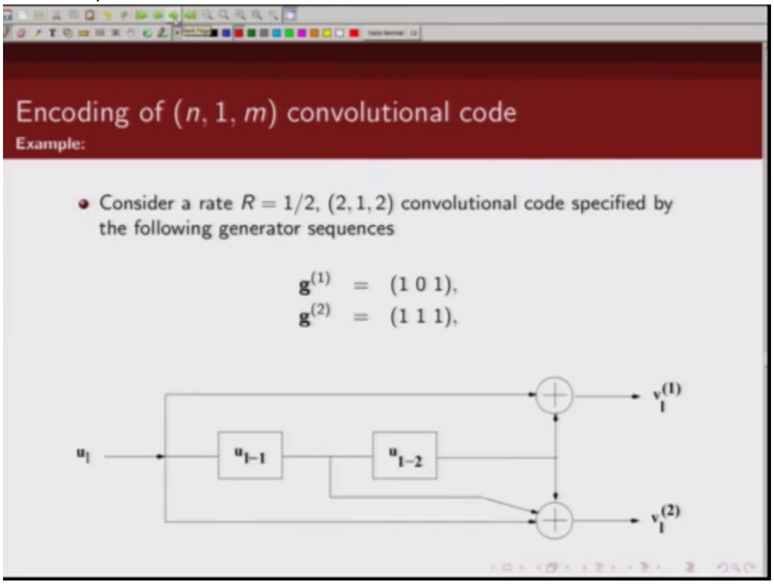

And what is my output then? My output is nothing but it is a discrete convolution of the information sequence with this generator sequence. So if my generator sequence, if my code has memory m, then basically I can write this discrete convolution in this particular fashion.

(Refer Slide Time 13:37)



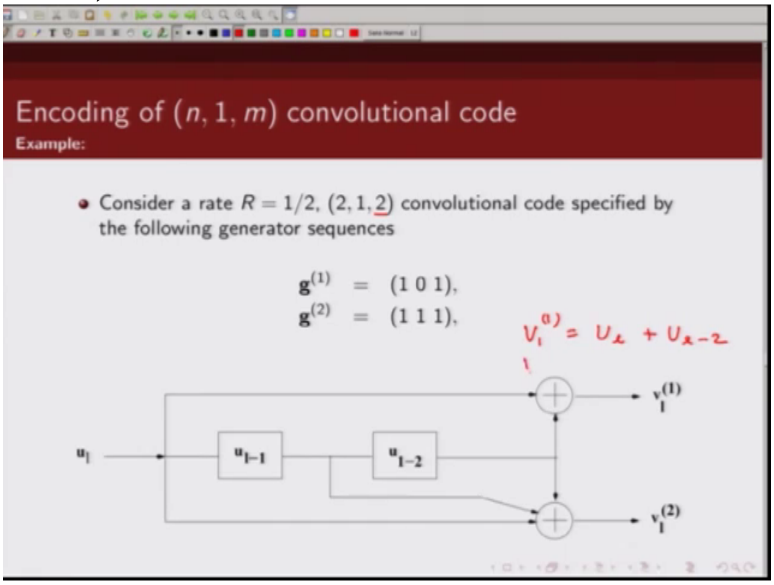

And that's basically my output sequence, which is discrete convolution of the input sequence with this generator sequence. Now let us take an example.

(Refer Slide Time 13:54)



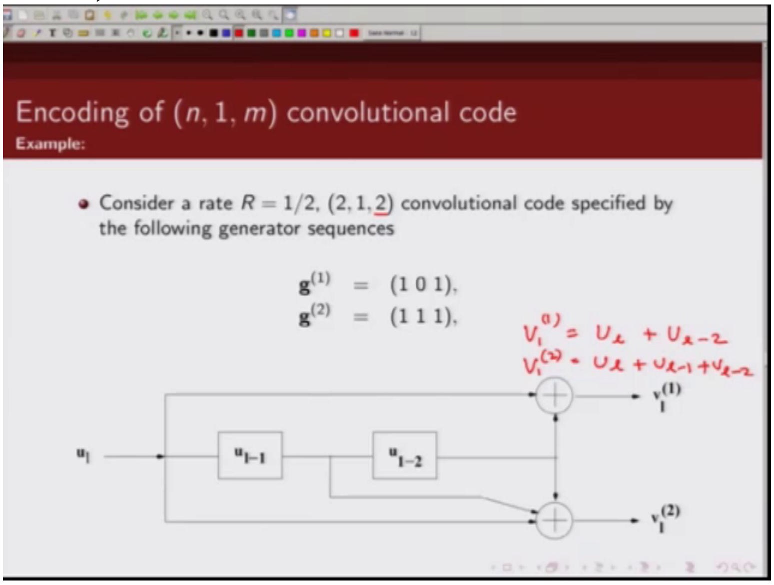

This is the same example that we are considering, this rate one half code with memory 2 so you can see v 1 l, this is basically again discrete convolution of input with this generator sequence which we can write as u l plus u l minus 2 and this

(Refer Slide Time 14:20)

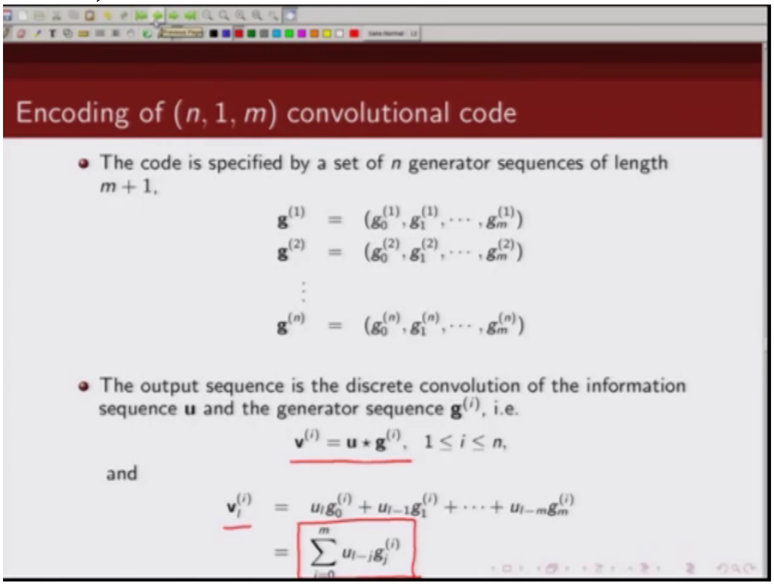



v 1 2 can be written as u l plus u l minus 1 plus u l minus 2. If we go back,

(Refer Slide Time 14:30)

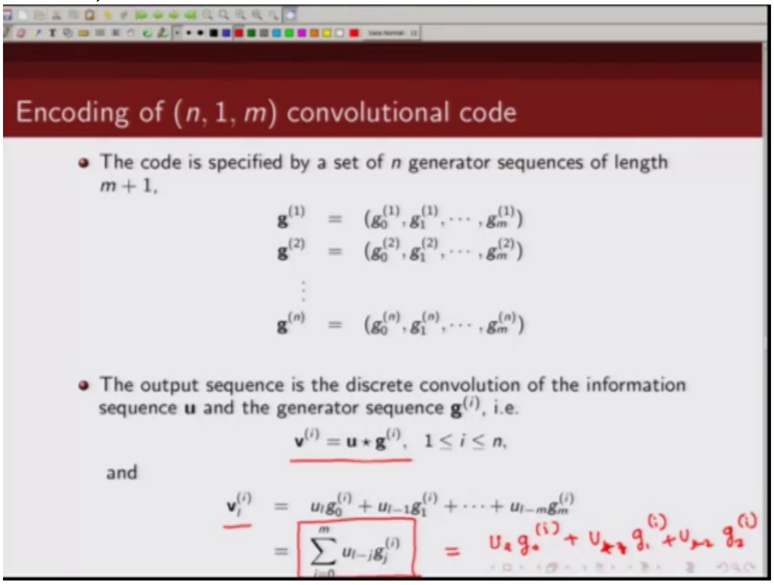



(Refer Slide Time 14:32)



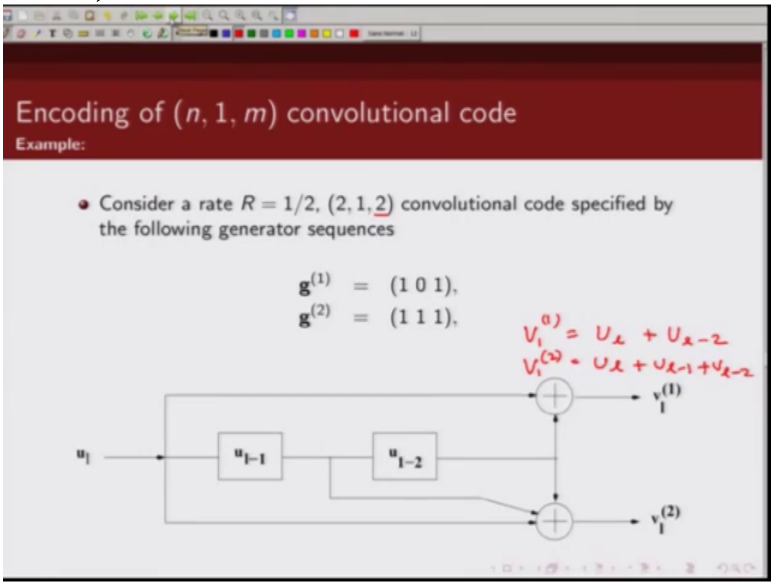

our output is this if we can expand it for this particular example this will be u l g 0 i plus u l minus 1, u l minus 1 g 1 i plus u l minus 2 g 2 i.

(Refer Slide Time 15:05)



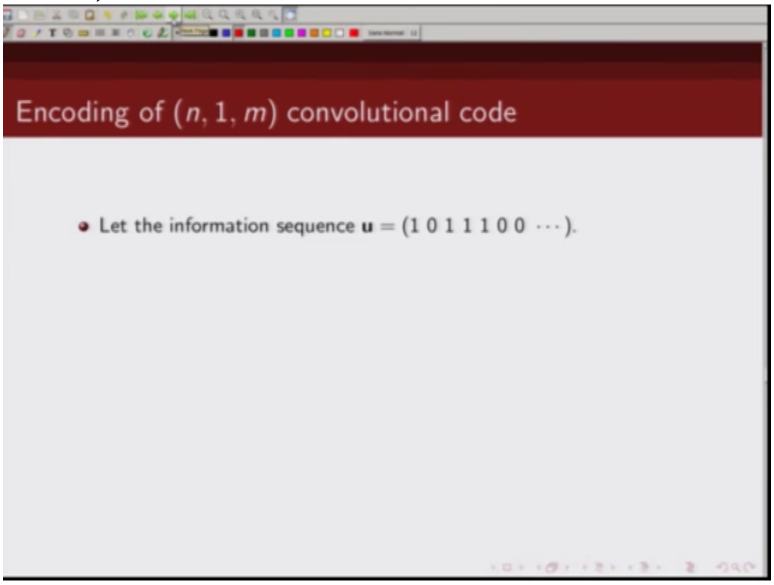

And for the first coded sequence, this g is, g 0 g 1 g 2 was 1 0 1 and the second sequence was 1 1 1, that's why the first

(Refer Slide Time 15:18)



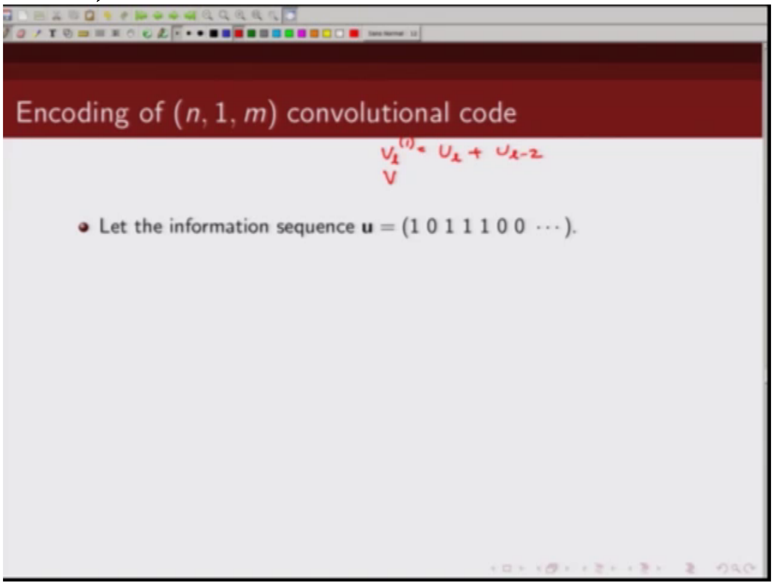

coded sequence is u l plus u l minus 2 and the second coded sequence is u plus u l minus 1 plus u l minus 2. So we

(Refer Slide Time 15:31)



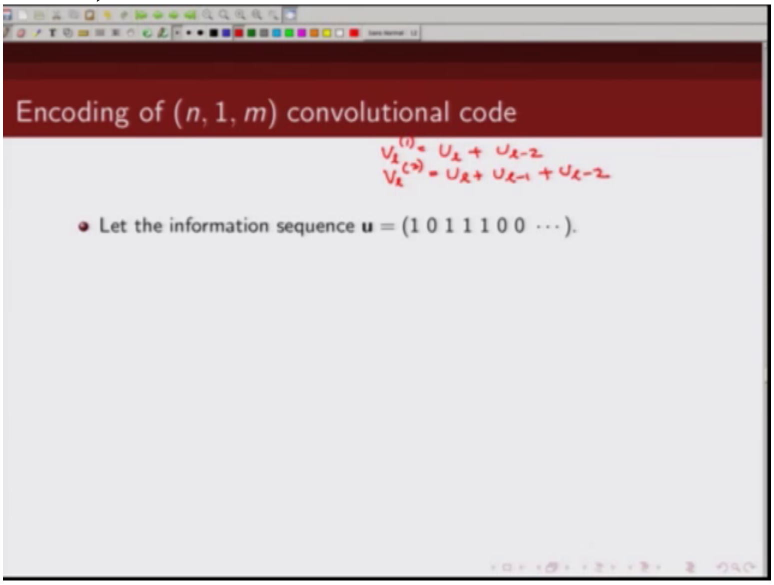

have information sequence this. What was our output sequence? We had v l 1 is u l plus u l minus 2 and

(Refer Slide Time 15:47)



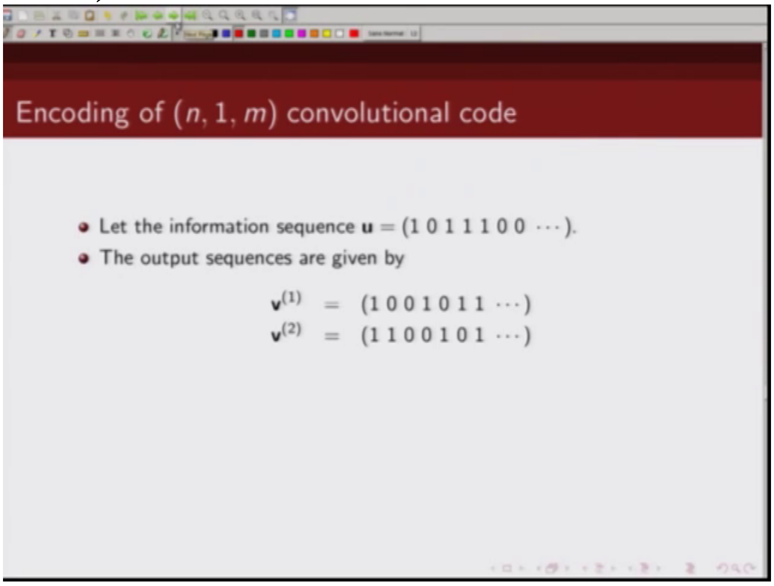

v l 2 is u l plus u l minus 1 plus u l minus 2. Now

(Refer Slide Time 15:56)



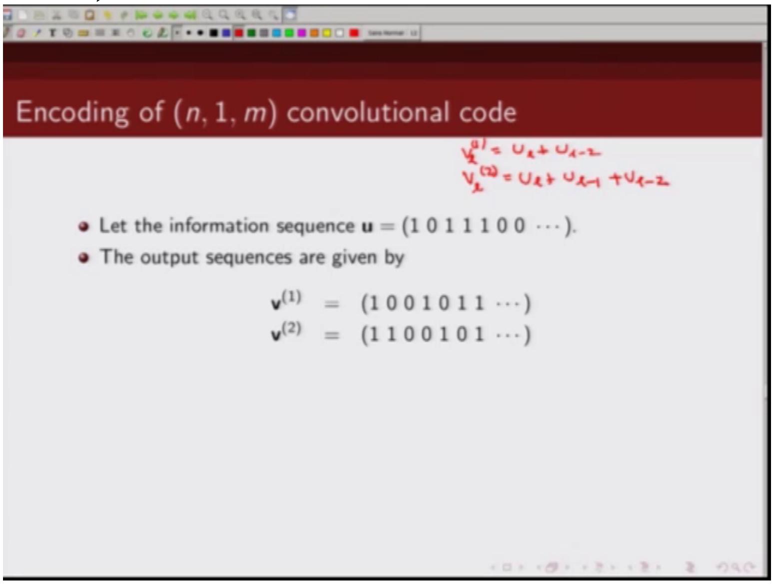

we can show that

(Refer Slide Time 15:58)



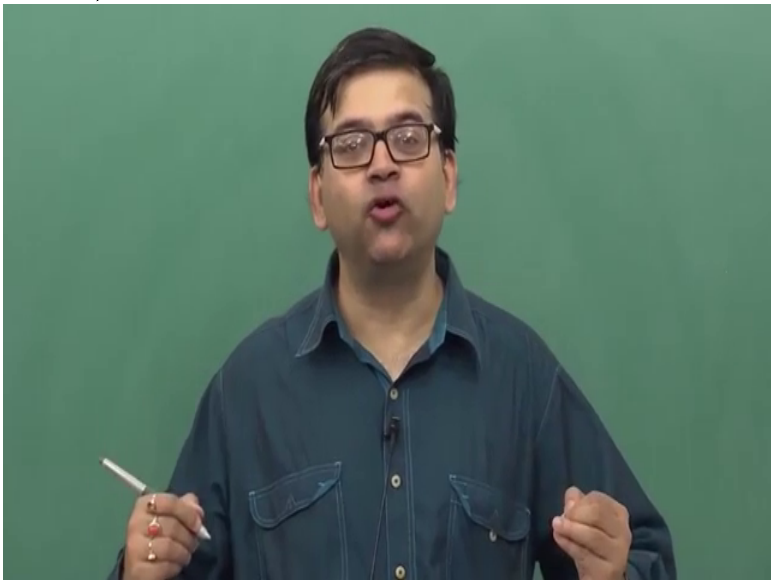

our output coded sequence will be given by this. Now this can be easily verified. So let's say what was our output coded sequence? v l was u l plus u l minus 2 and v, v l 2 is u l plus u l minus 1 plus u l minus 2. Now note

(Refer Slide Time 16:24)



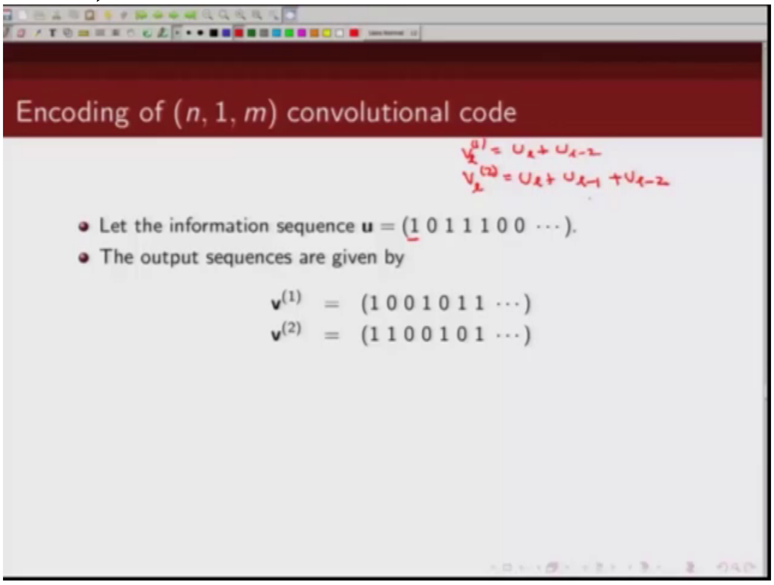

when the first input u l which is 1 comes, what is the output? Now to specify the output we need to specify what the initial contents of u l minus 1 and u l minus 2. So initially we will assume that the

(Refer Slide Time 16:44)



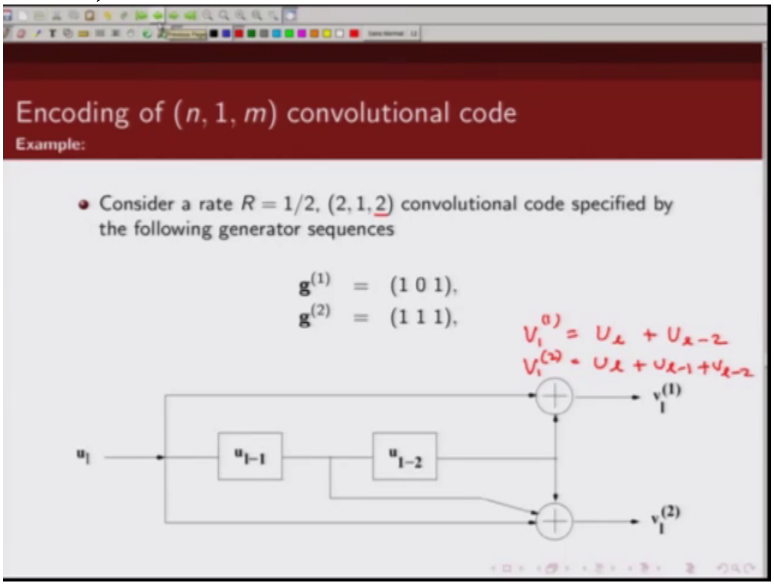
convolutional encoder was in all zero state. What do we mean by all zero state? So we are assuming that initially the contents of the shift register were all zero. In other words u l minus 2 and u l minus 1, they were both zero, Ok. If both were zero initially and if u l is 1, then what will be v l 1?

(Refer Slide Time 17:09)



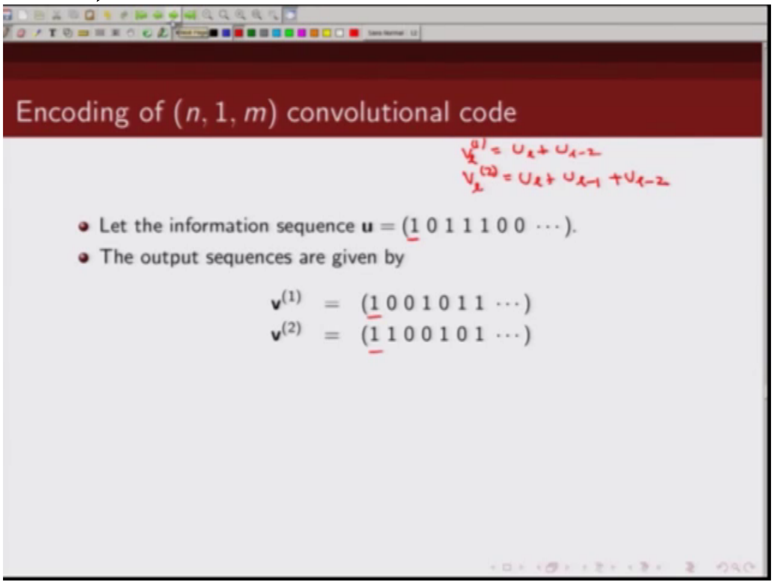

This is 1 plus 0 which is 1, so you can see this is 1. And what is v 1 2? This is 1 plus 0 plus 0, so that's also 1. Next what happens, next if you go back this 1 which was here,

(Refer Slide Time 17:33)



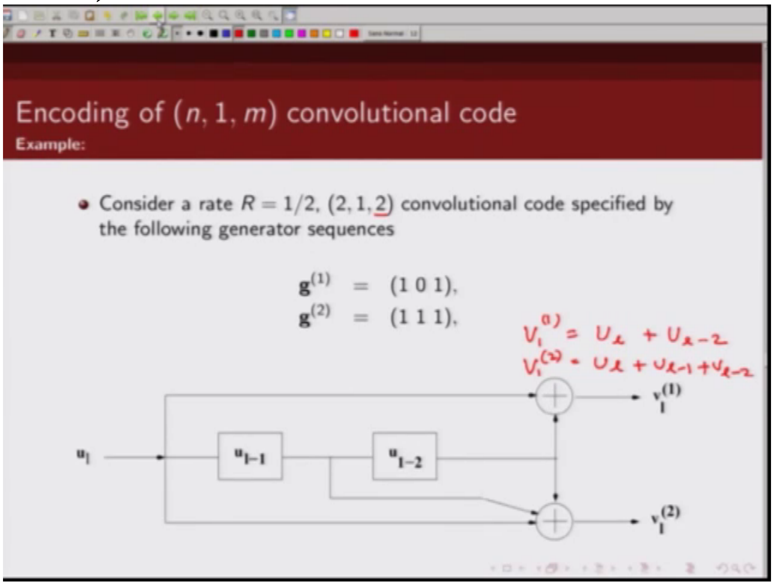

when you apply a clock this 1 moves here and a new bit comes here. So now the next time instance u l minus 1 becomes 1 and what is u l minus 2? Since u l minus 1 initially was zero so this zero will come here. So the new contents of the shift register will be now 1 and 0. So

(Refer Slide Time 17:58)



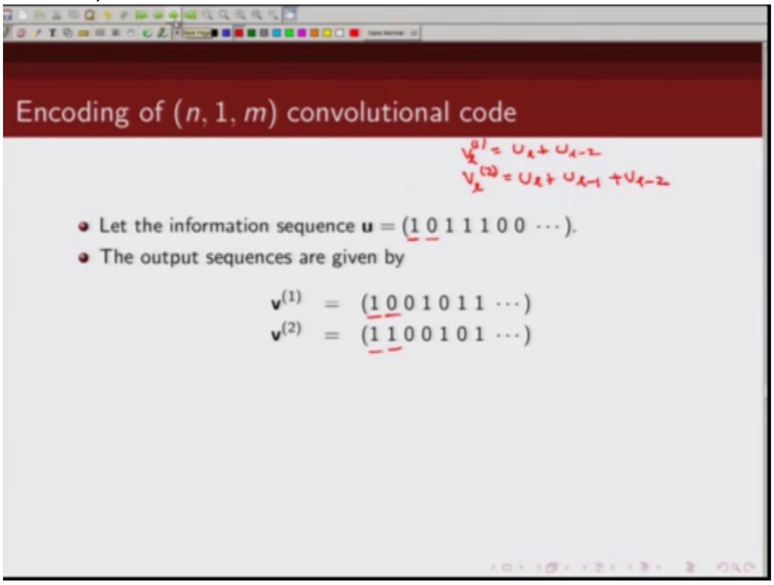

what we have is now u l minus 1 is 1 and u l minus 2 is 0. Now the next input is zero. The next input is zero so the what's the next output? This is zero and u l minus 2 is zero so this will be zero. You can see this is zero. What about this? Now u l is zero, u l minus 1 is 1, and u l minus 2 is zero. So zero plus 1 plus 0, that will be 1 and that's given by this, Ok. Next what happens? Again go back to

(Refer Slide Time 18:39)



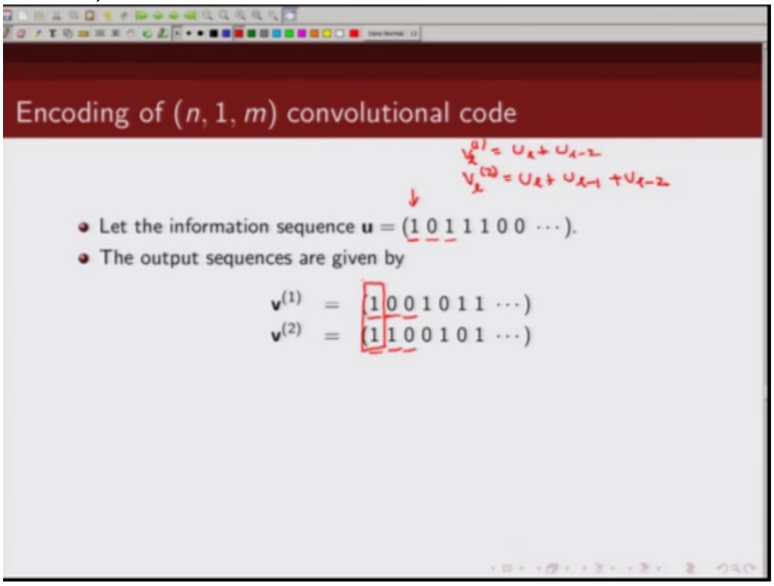

this diagram. You had input zero here. So now this zero will move here and you had a 1 here. So this 1 will move here. So the new content of the shift register will be zero and 1, Ok. If that happens then next

(Refer Slide Time 18:58)



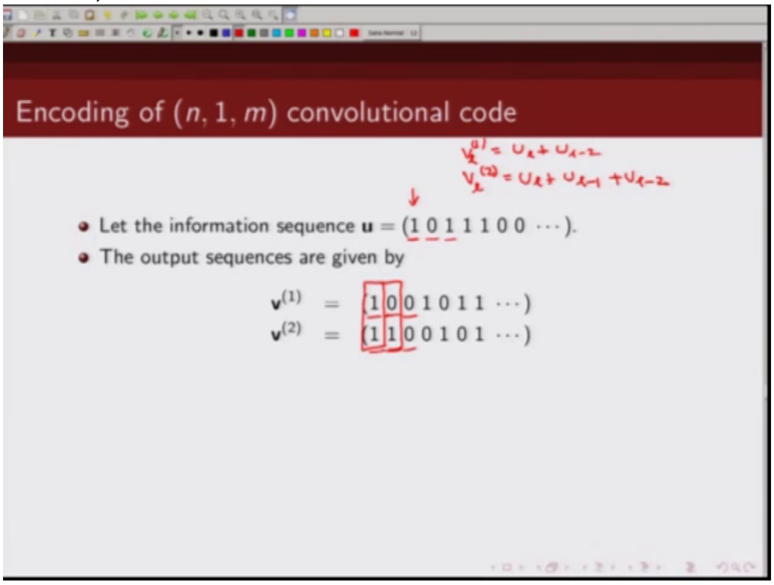

input is 1. So this is 1. What is u l minus 2? u l minus 2 was 1 so 1 plus 1 that's zero. And u l is 1, u l minus 2 is 1 and u l minus 1 is zero. So 1 plus zero plus 1 that's zero. So like that, you can basically write down the output coded sequence. So then what is my final output? So corresponding to this 1 and what is my coded sequence? That's given by this.

(Refer Slide Time 19:39)



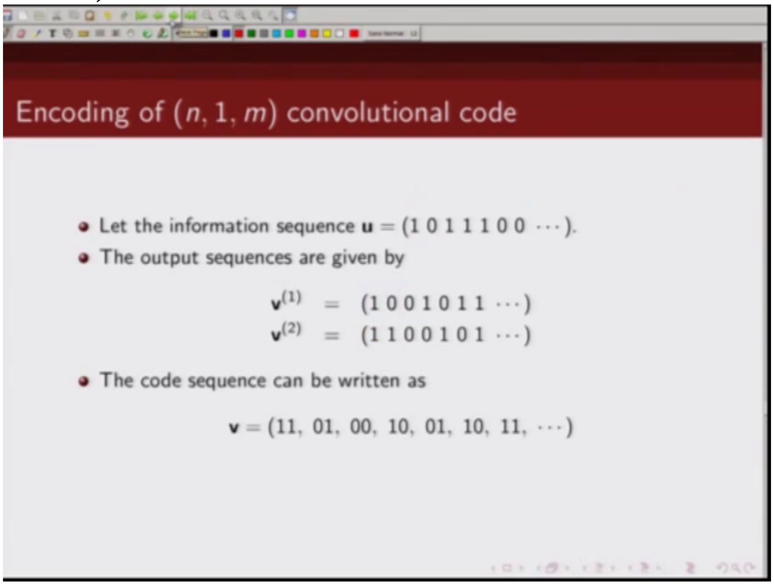

Corresponding to this zero, my coded sequence is given by this,

(Refer Slide Time 19:45)



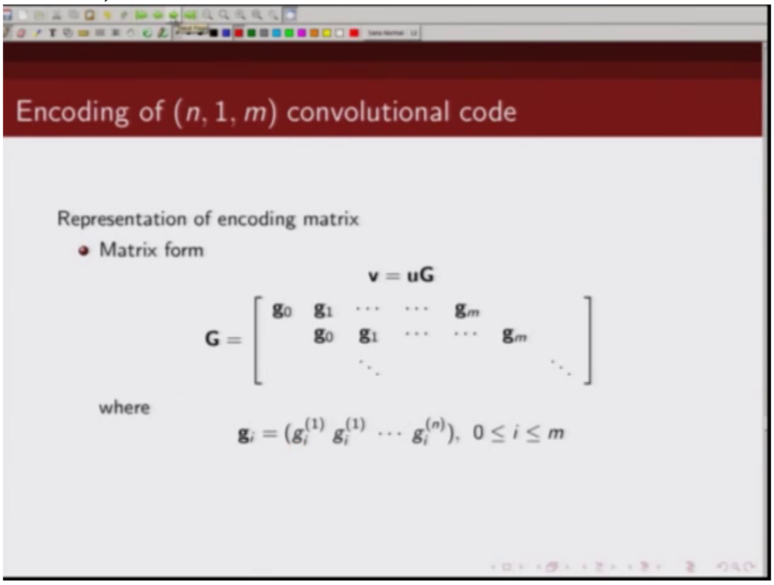

Ok. So then I can write my

(Refer Slide Time 19:48)



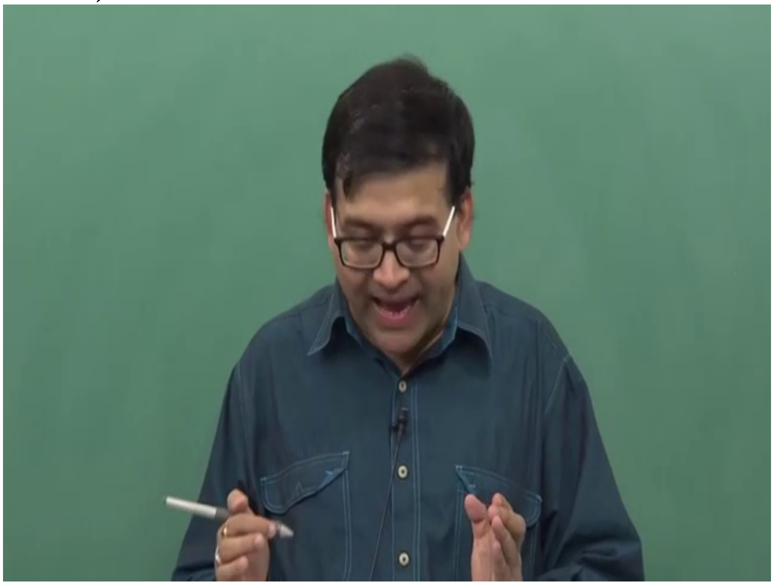

final output as, corresponding to input 1, I get 1 1. That's given by this. Corresponding to zero I get 0 1, that's given by this. Corresponding to 1, I get 0 0, that's given by this. So this is how I can write my output coded sequence.

(Refer Slide Time 20:12)



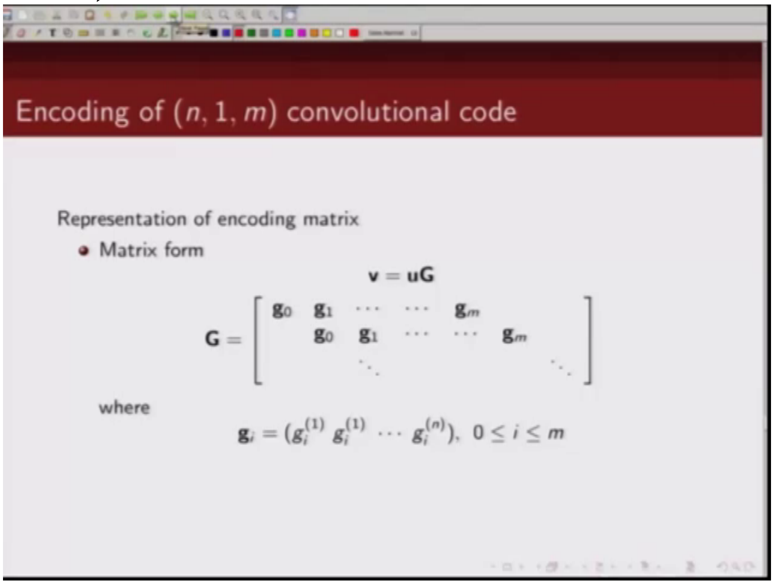

Now the same thing I can write in the

(Refer Slide Time 20:15)



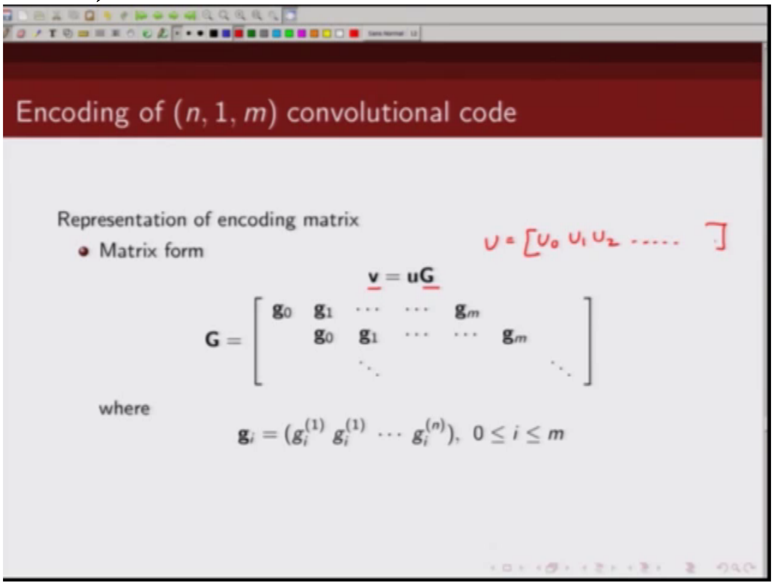
matrix form. So I define this generator

(Refer Slide Time 20:19)



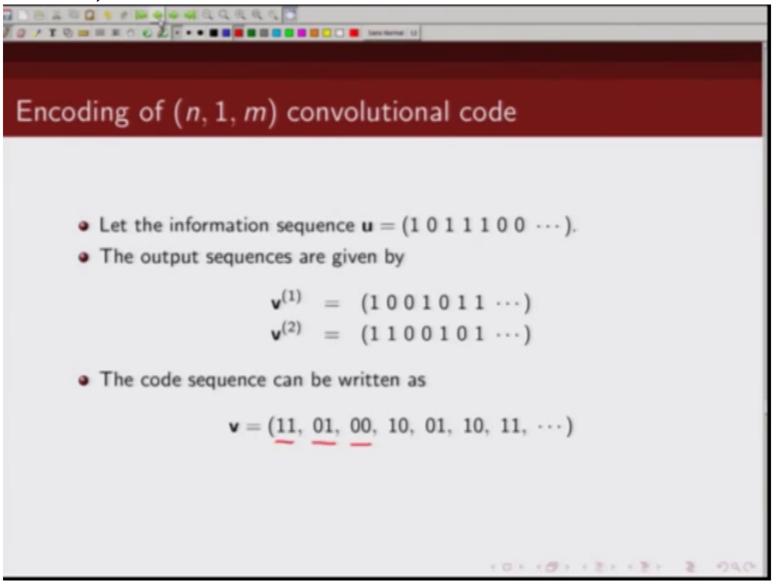

matrix G which generates this codeword. So the output codeword can be written as input times this generator matrix G, Ok and this generator matrix is of the form like this. So let's just expand it and may be try to explain why the generator form has this semi infinite kind of form for a convolutional code. So let's say u is u 0, u 1, u 2 dah dah dah dah it's continuing set of

(Refer Slide Time 20:58)



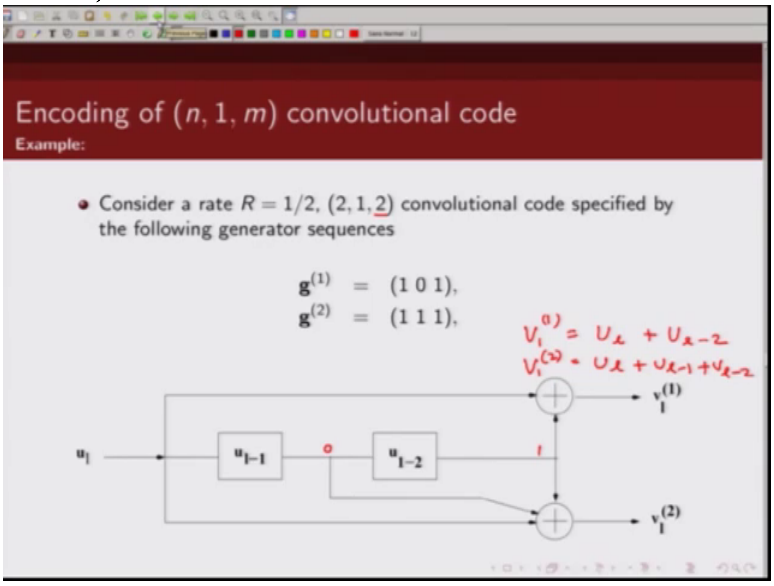

sequence like this right now what is your output sequence? Output sequence so initially what happens, if you go back to this diagram,

(Refer Slide Time 21:12)



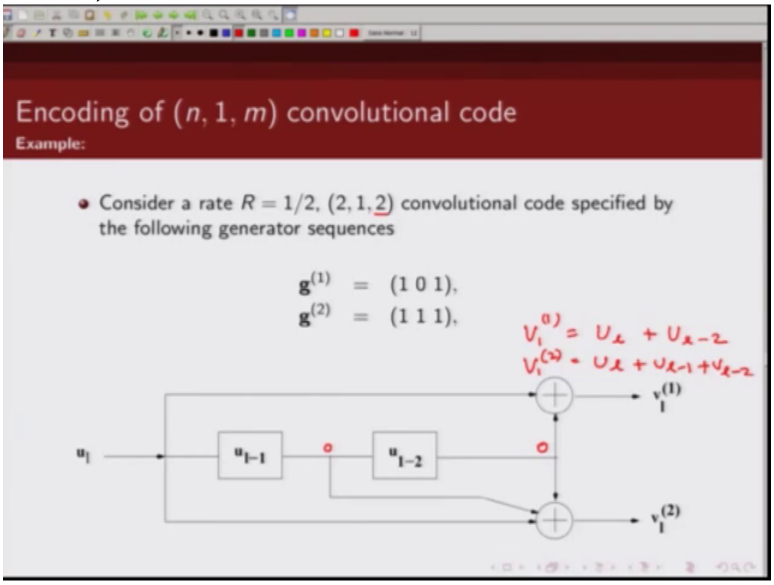

initially you are assuming

(Refer Slide Time 21:17)



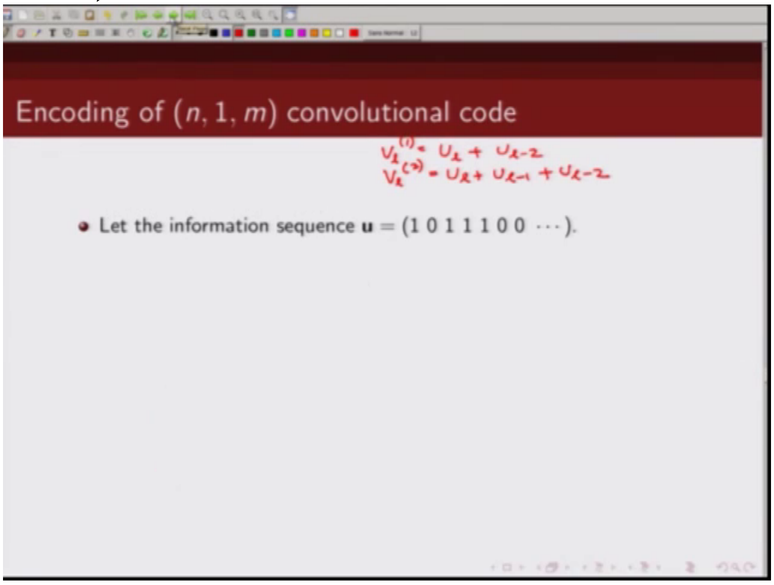

that the encoder is in all zero state

(Refer Slide Time 21:26)



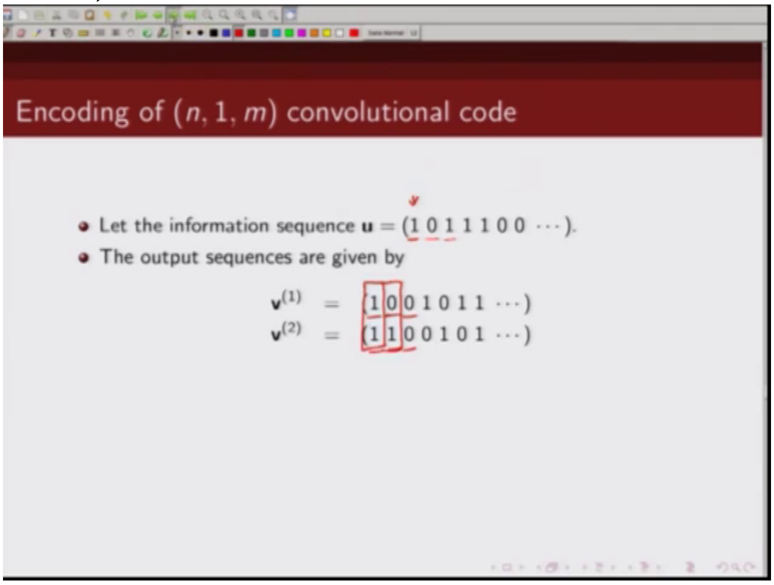

correct? So what will be the first output that you will get here? That is nothing but u 0 times g 0. What is g 0? g 0 1 is this, g 0 2 is this, this interconnection, which is connecting u i to the output. So at first time instance what you would get

(Refer Slide Time 21:51)



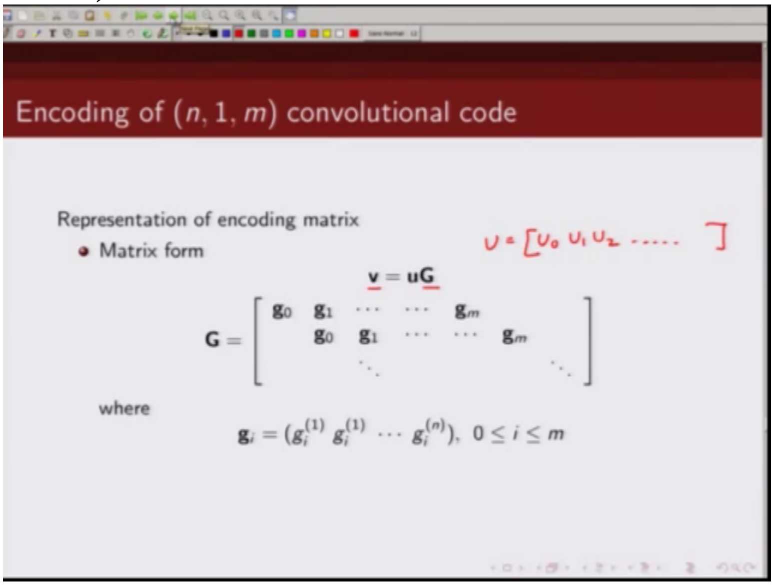

is the output is nothing

(Refer Slide Time 21:53)



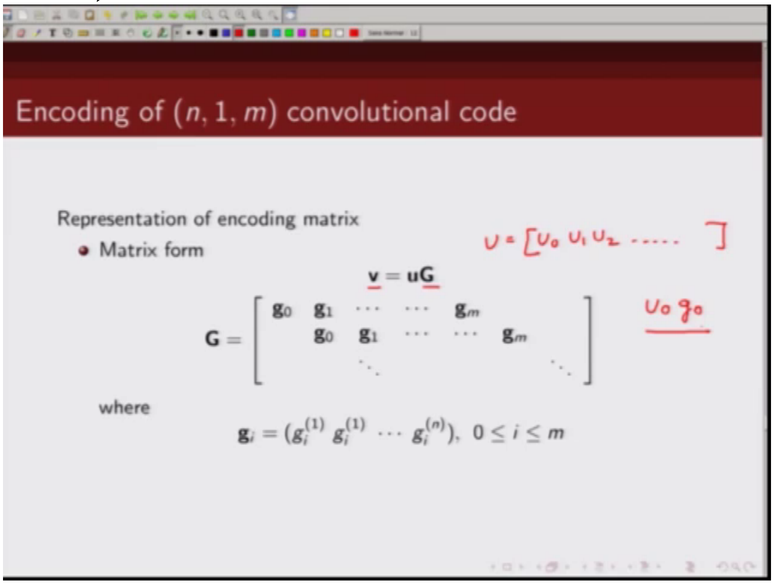

but

(Refer Slide Time 21:55)



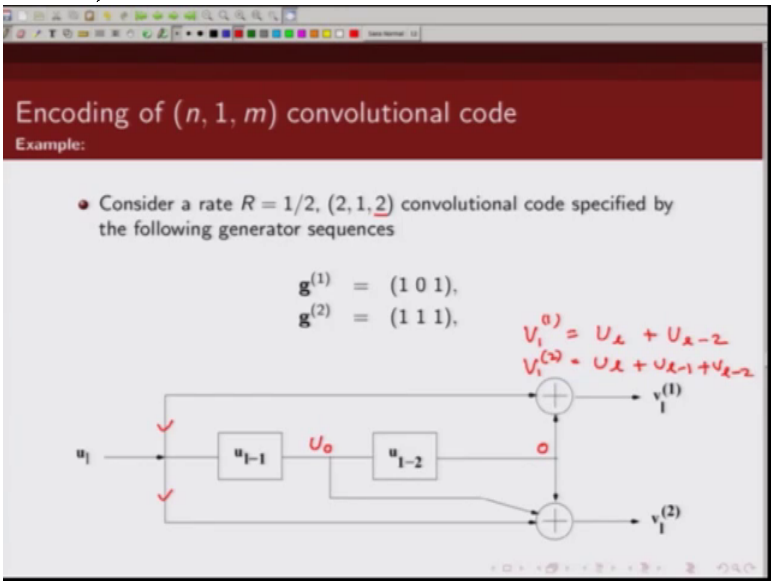

but u 0 times g 0. This is the output that you will get at first time instance.

(Refer Slide Time 22:01)



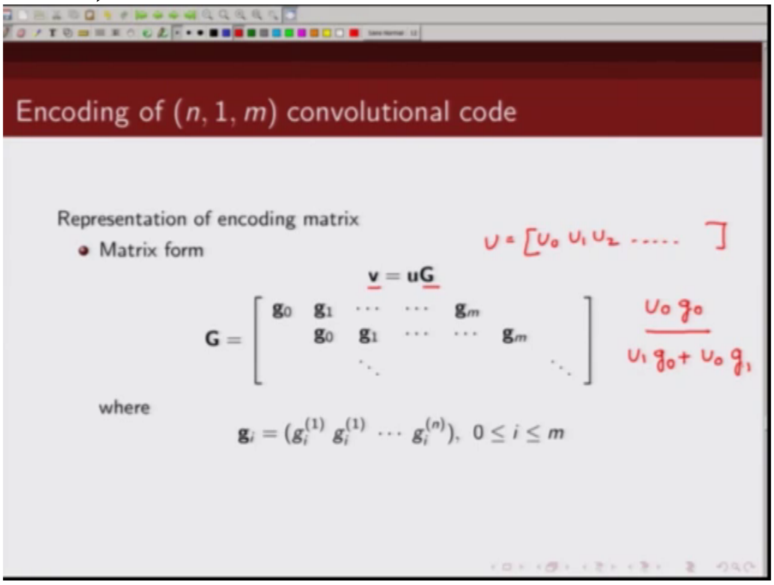

What is the output you will get at the second time instance? Now whatever u 0 you had, now that u 0 has moved here,

(Refer Slide Time 22:12)



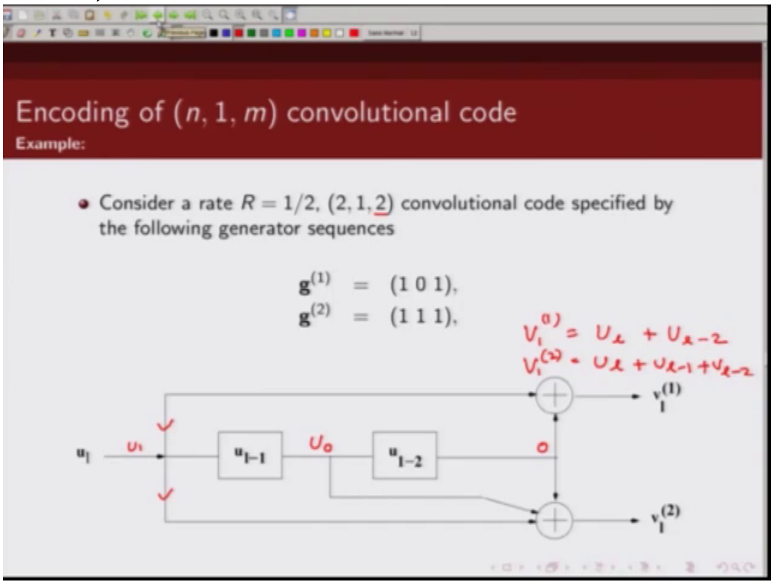

correct and a new bit which is u 1 has come here, u 1. So what is the output at this time? It is u 1 times g 0 plus u 0 times g 1.

(Refer Slide Time 22:46)



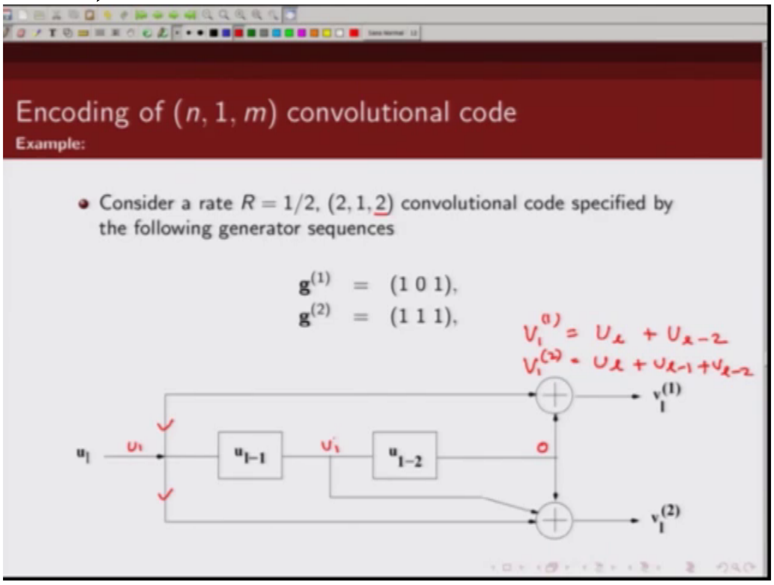

So I can write, and the second time instance my output is given by u1 times g 0 plus u 0 times g 1, fine. Next time instance, what is my output? Again go back, now what is going to happen is,

(Refer Slide Time 22:58)



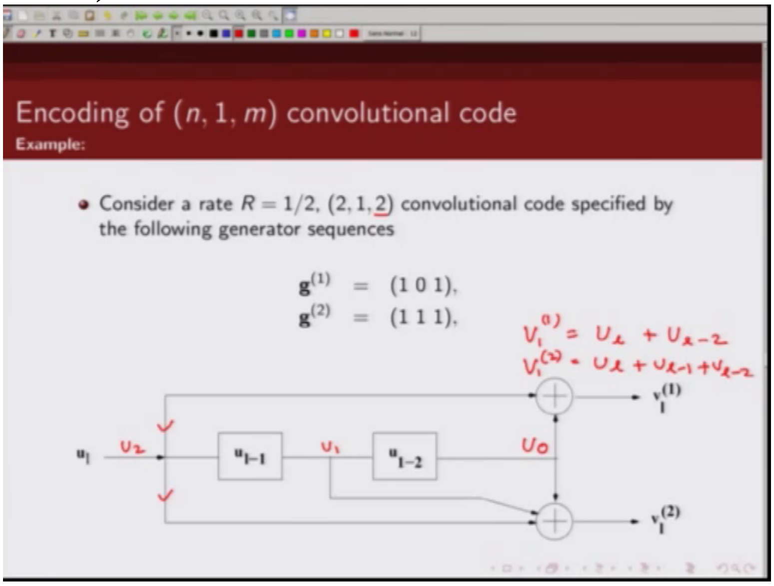

this u 1 will move here. So this will be now u 1.

(Refer Slide Time 23:07)



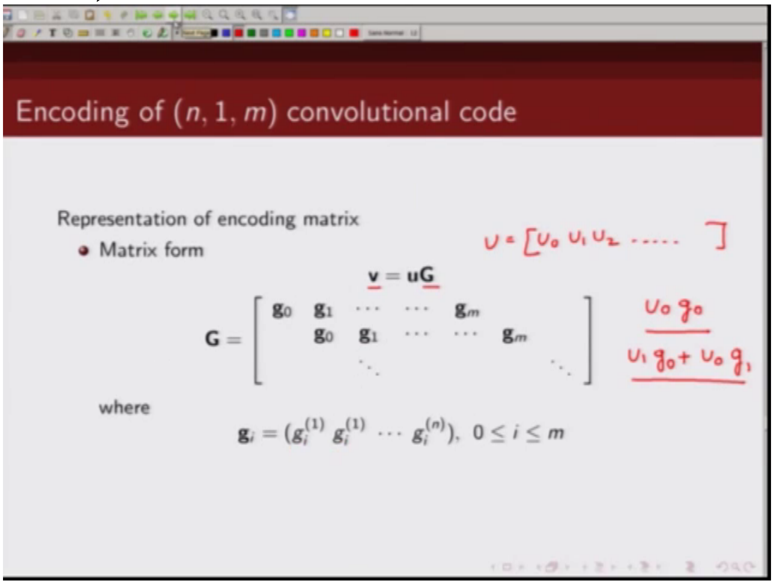

This will become u 2 and this will become u 0. So what is my output now?

(Refer Slide Time 23:21)



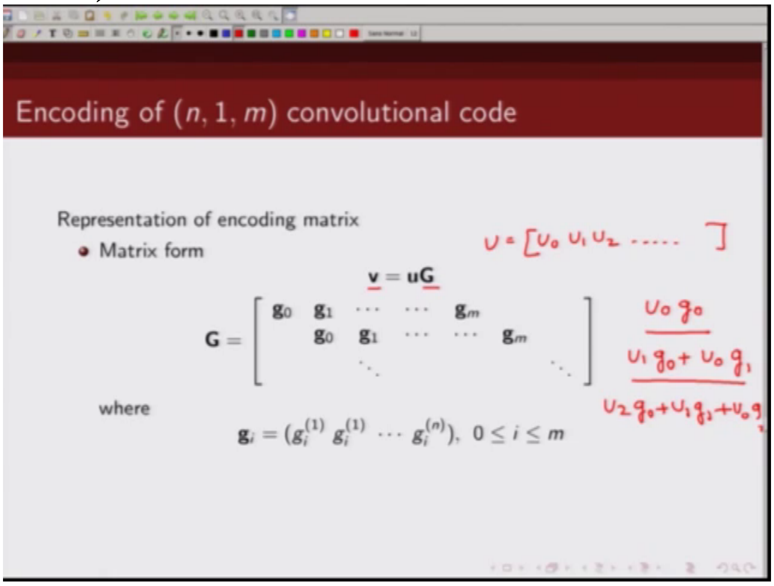

It is u 2 times v 0 plus u 1 times v 1 plus u 0 times g 2. So go back, so what would

(Refer Slide Time 23:40)



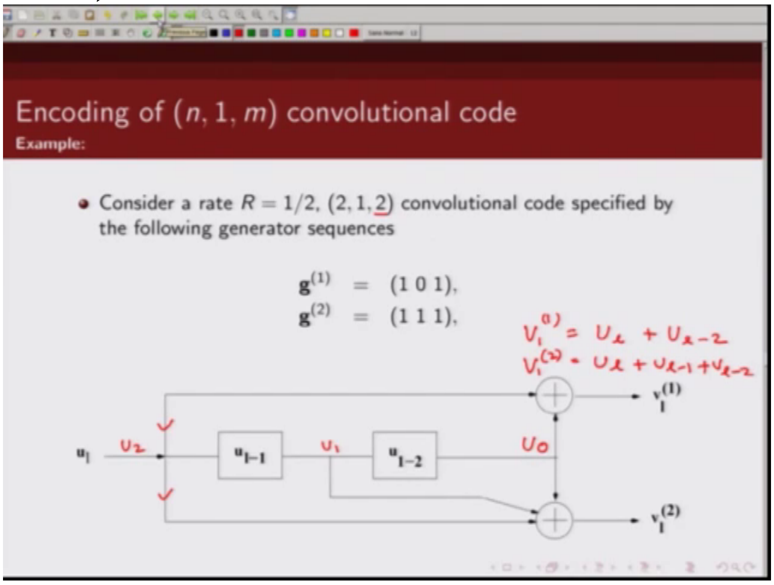

be my output here? It is u 2 times v 0 plus u 1 times g 1 plus u 0 times g 2.

(Refer Slide Time 23:54)



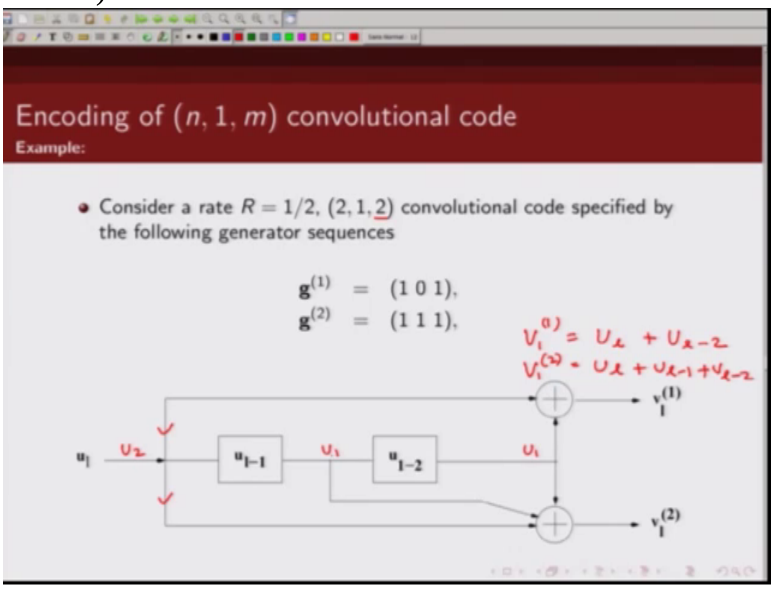

What happens next?

(Refer Slide Time 24:00)



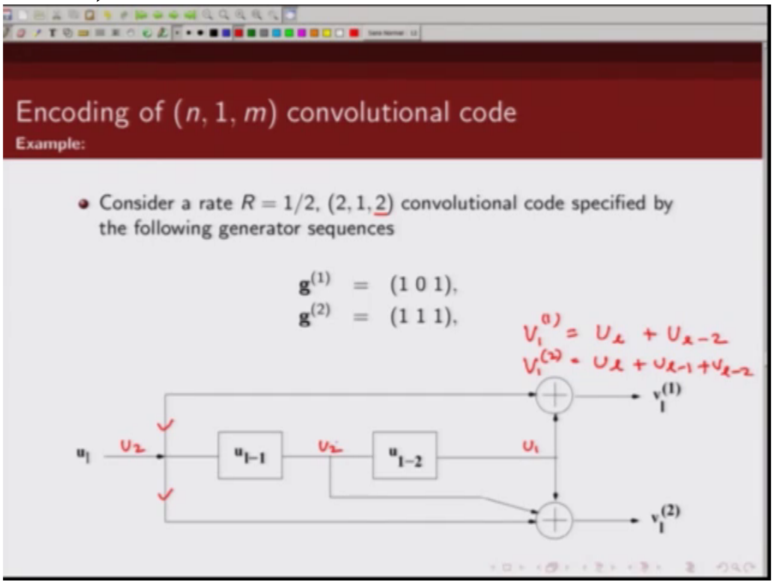

This u 0 moves out. Here what we will get is u 1, this will be u 1. What about this,

(Refer Slide Time 24:14)



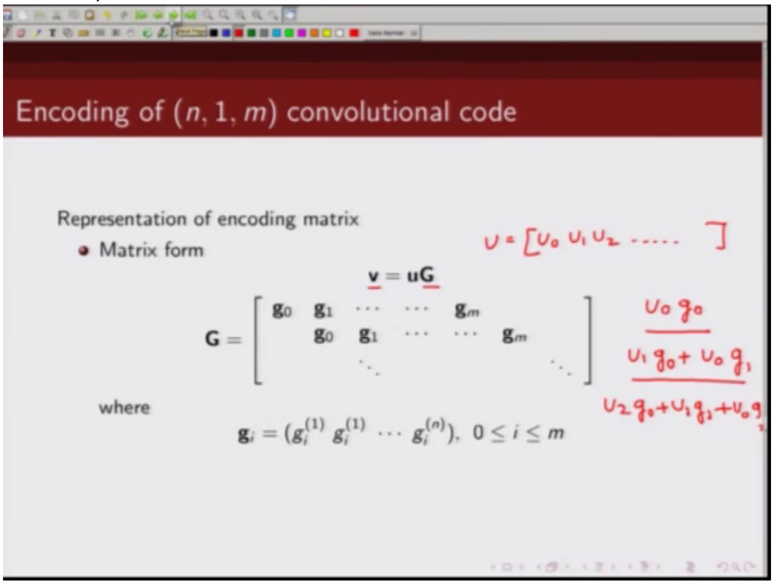

this will become u 2, so this is u 2 and this will

(Refer Slide Time 24:21)



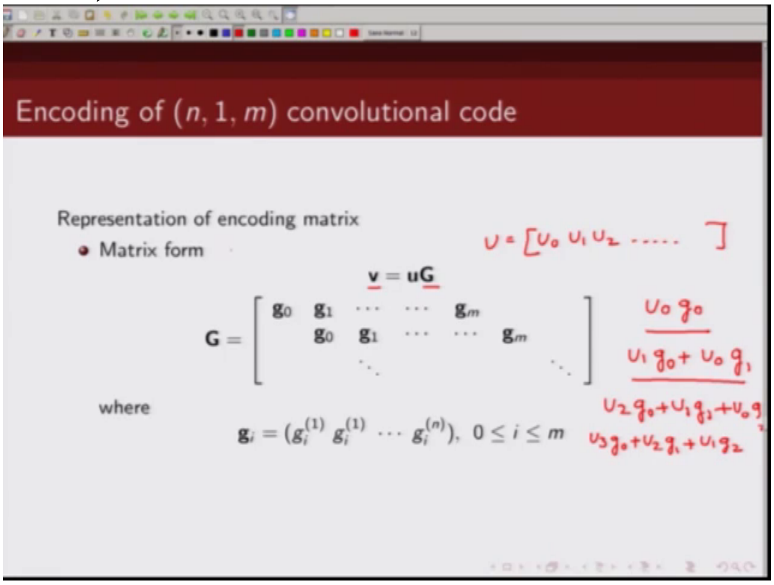

become u 3. So this is u 3. This is u 3. So what will be the output now? Its u 3 times g 0 plus u 2 times g 1 plus u 1 times g 2. And u 0 does not appear because the memory order of this code was 2. So what is the output in this case? Third instance, this will be

(Refer Slide Time 24:48)



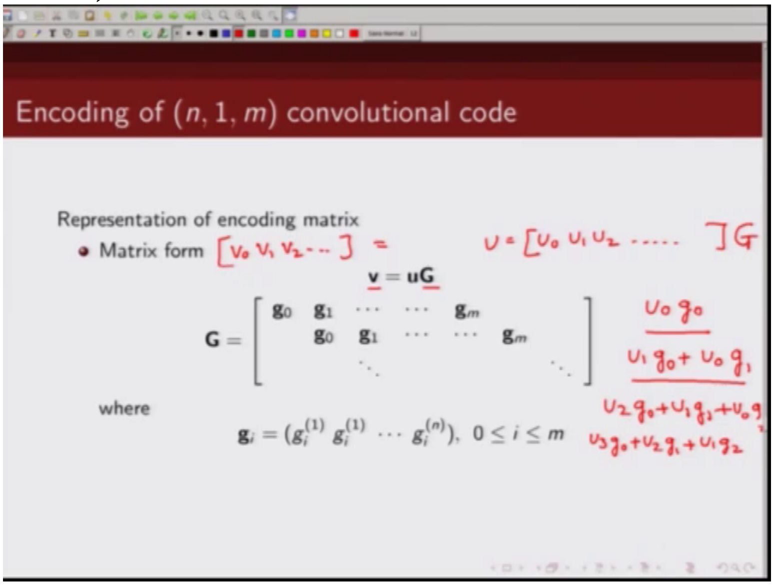

u 3 times g 0 plus u 2 times g 1 plus u 1 times g 2. Now if we write the same thing in a matrix form, so what is v? v is

(Refer Slide Time 25:07)



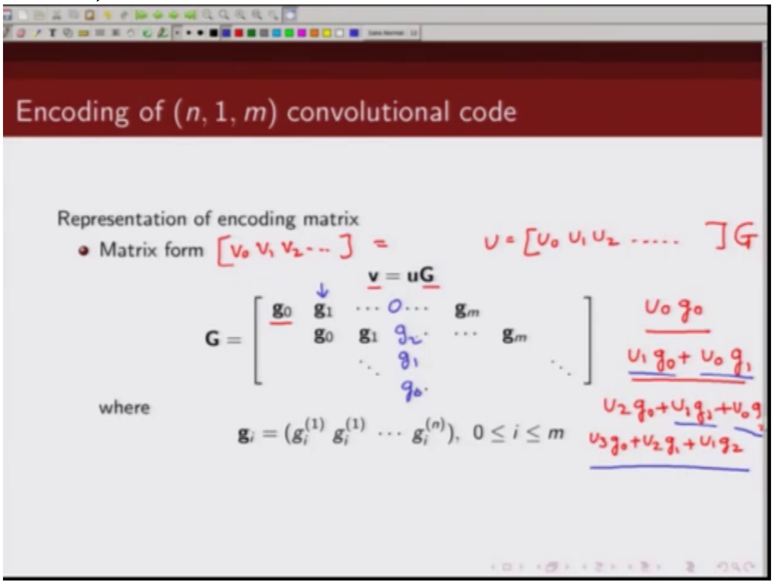

basically, v at times zero, time 1, time 2, if we write this in this particular form, is equal to u times this matrix G.

(Refer Slide Time 25:17)



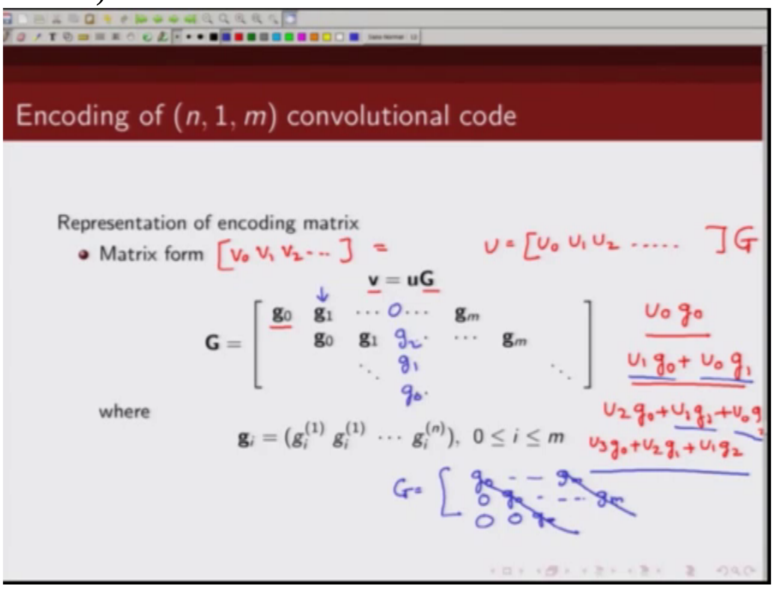

Now you compare this equation with this equation. So at first time instance, output is u 0 g 0. So that's what, this is u 0 times g 0. So this is g 0. Second instance, what is my output? My output is u 0 times g 1, this term, u 0 times g 1 is this term and then g 0 times u 1, which is this term. So the second entry of this generator matrix is this, Ok. Now what's the third entry you can see? u 0 times g 2, so u 0 times this is g 2, plus u 1 times g 1, so this is g 1 and then this is u 2 times g 0. So you can see. Then further if we look at this, what we get here is, so u 0 times zero will get here and then and we will get u 1 times g 2, u 3 times so this will be like zero and g 2, g 1 g 0 so in this case the memory order was m. That's why we are getting like this.

(Refer Slide Time 26:37)



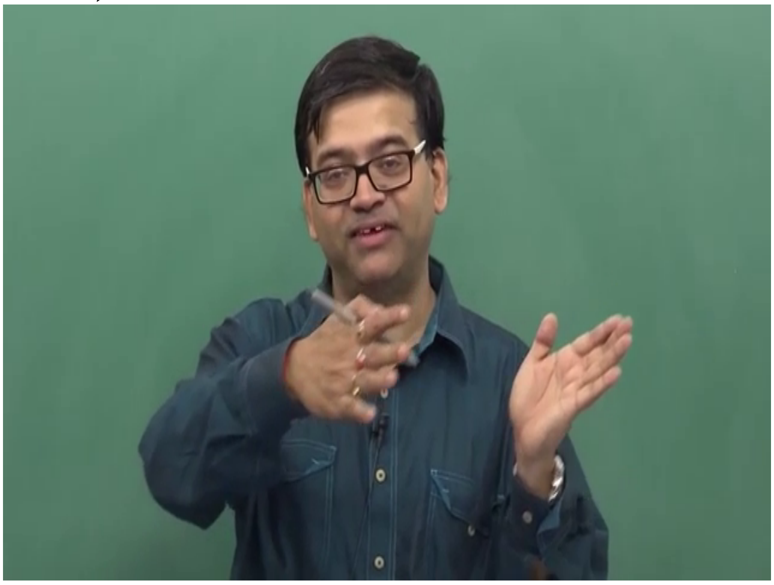

So you can see here, our generator matrix is of the form, of semi infinite form where basically our G is something like this, so we have g 0 to g m, now this becomes 0, now this is u g 0 and this is g m and this 0 0 and this is g 0, so it is like in this way diagonally my

(Refer Slide Time 27:01)



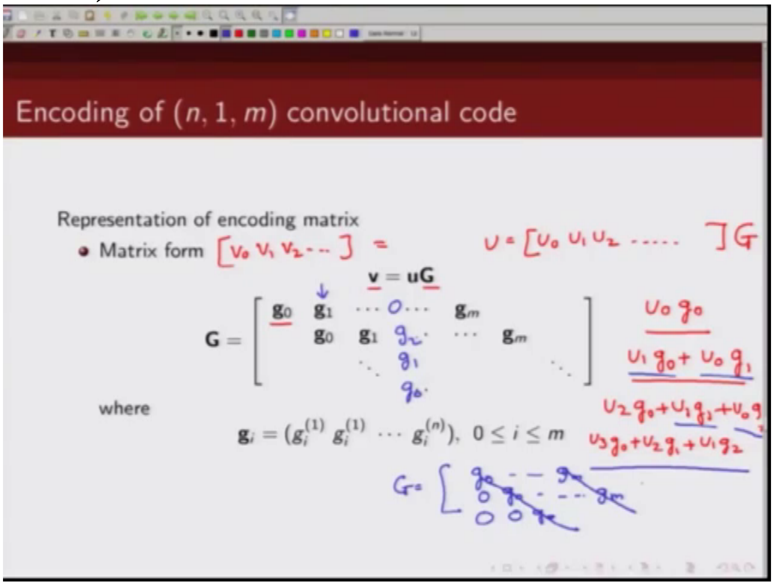

generator sequence is moving. And that's what I have written here. So if I try to write it in the form of generator matrix then I can, my generator matrix in this case is a semi infinite

(Refer Slide Time 27:14)



form and through this example for a memory 2 code

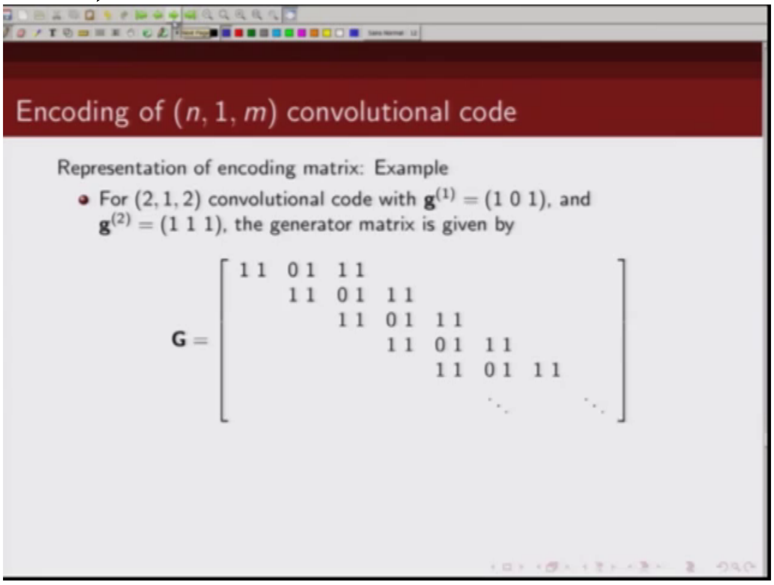
(Refer Slide Time 27:18)




we showed that this specifically G is of the form like this, Ok. And where each of these g 0's are basically these, will represent what are these n bit output.

So let us

(Refer Slide Time 27:35)

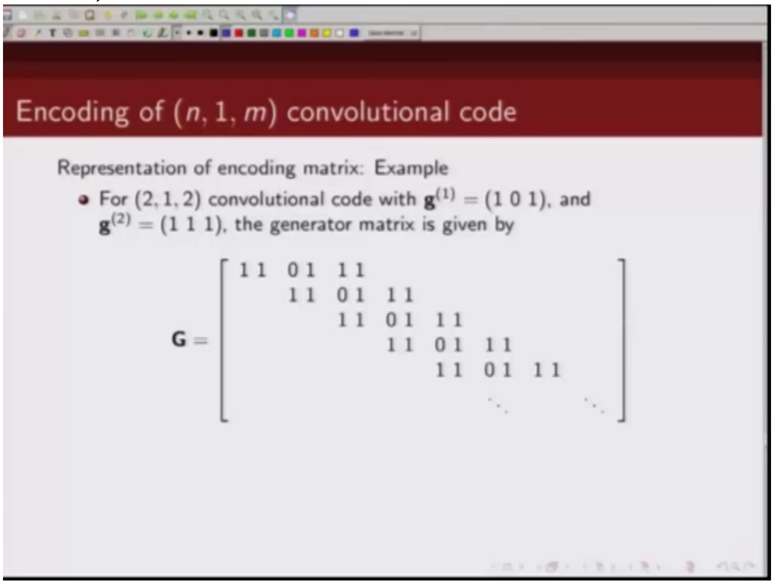



continue with the example that we are

(Refer Slide Time 27:37)

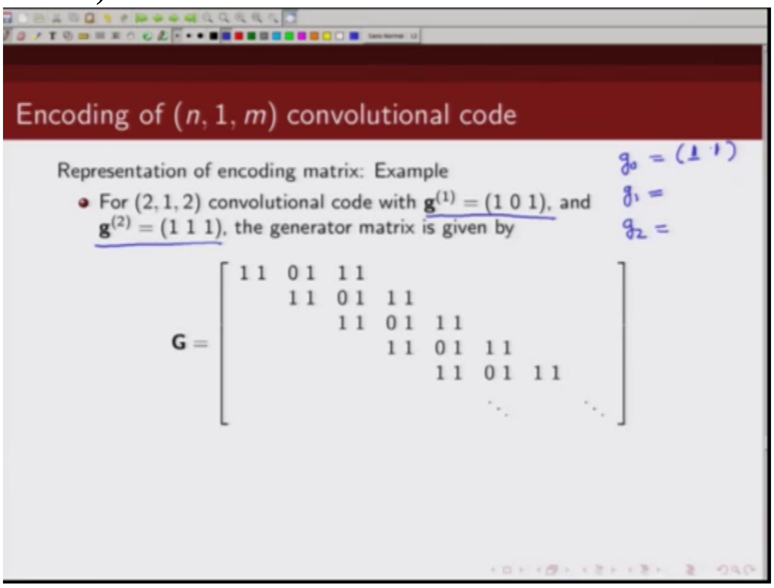
considering so far. So we are continuing with our rate 1 by 2 code whose memory order is 2 and we know our generator

(Refer Slide Time 27:47)



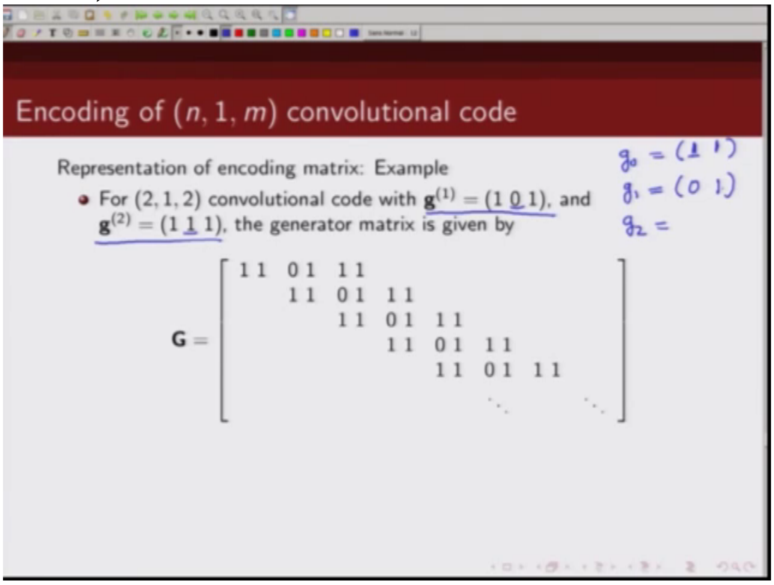

sequence. For the first code, the sequence is given by 1 0 1, because my output v l is u l plus u l minus 2. Similarly the generator sequence for the second codeword is given by 1 1, Ok. Then can I write basically what is my g 0, g 1 and g 2. So g 0 is given by, now there are 2 outputs so g 0 will have 2 terms, the first term corresponding to the first coded sequence so here this is 1, and what about the second coded sequence that is 1, so g 0 is 1 1, g 1 is this is 0,

(Refer Slide Time 28:37)



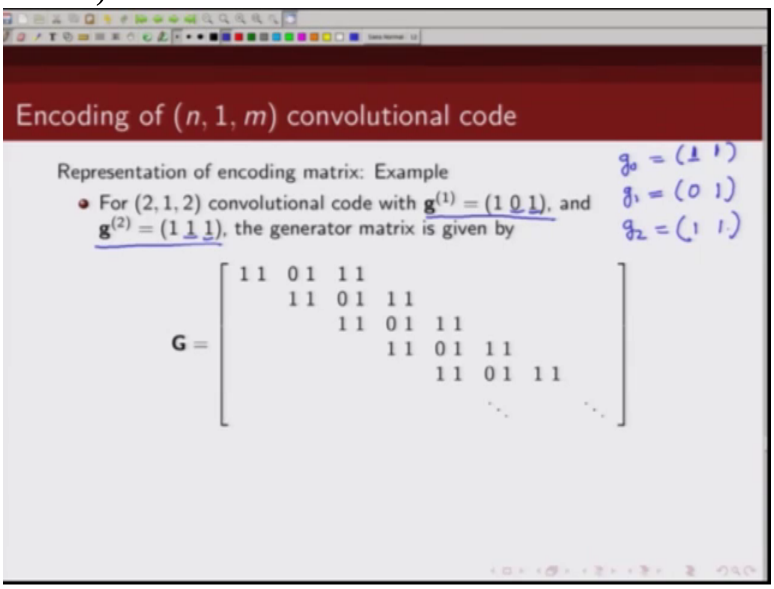

so this is 0, this is 0 and this is 1. So g 1 is 0 1.

(Refer Slide Time 28:44)



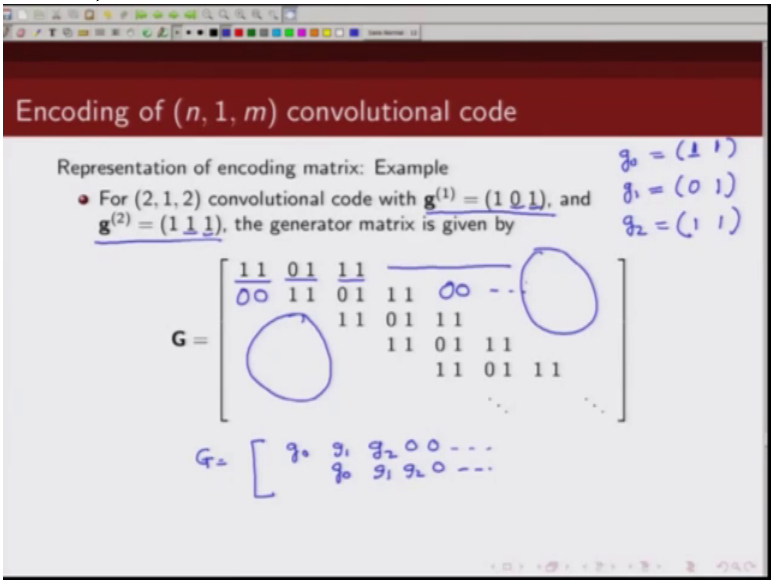

What about g 2? g 2 is this is 1 and this is 1. So g 2 is 1 1.

(Refer Slide Time 28:53)



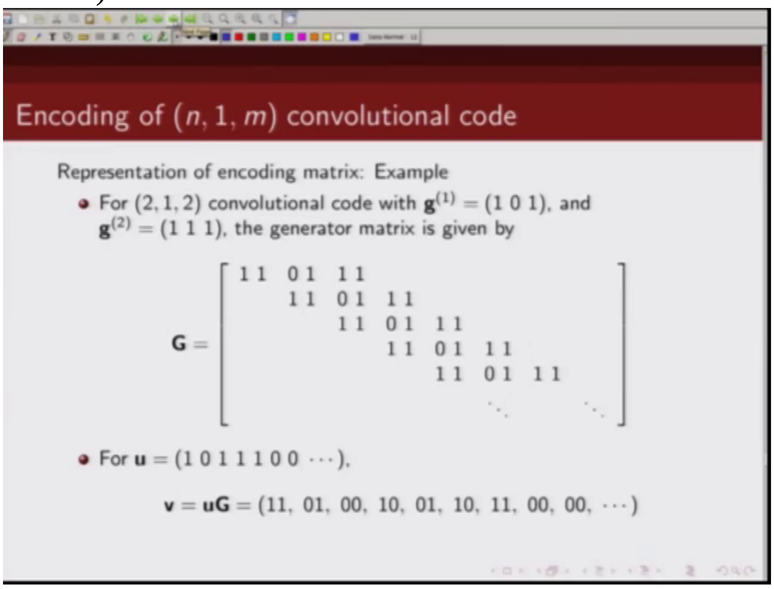

So I can then write my generator matrix which is of the form e is the form g 0 g 1 g 2 and the rest of all of these are basically zero. These are zero, these are all zero. This is g 0, g 1, g 2 and then these are all 0s. So what is g 0? g 0 is 1 1, that's what I have written here. g 1 is 0 1 that's what I have written here. And g 2 is 1 1, rest all these entries are 0. Similarly this is 0 0, and then I have g 0, g 1, g 2 and then these are all 0’s, Ok. So this is how I can write a

(Refer Slide Time 29:38)



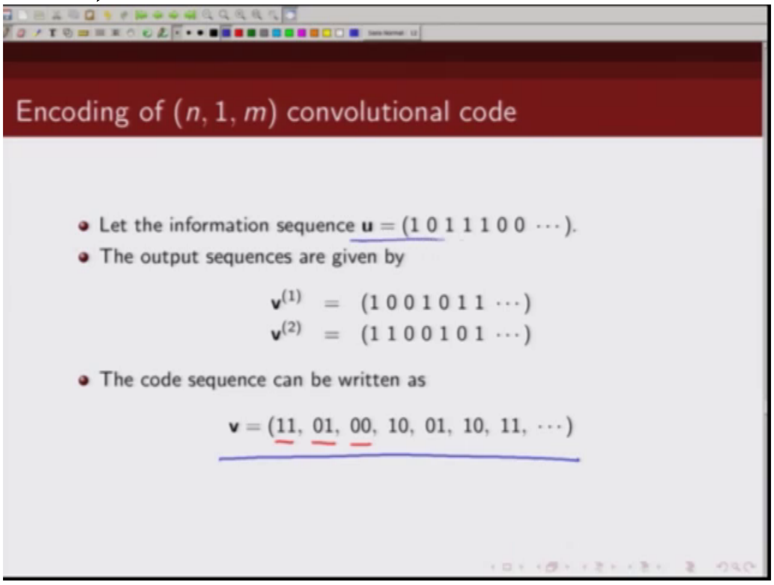

generator matrix. Now let's verify

(Refer Slide Time 29:42)



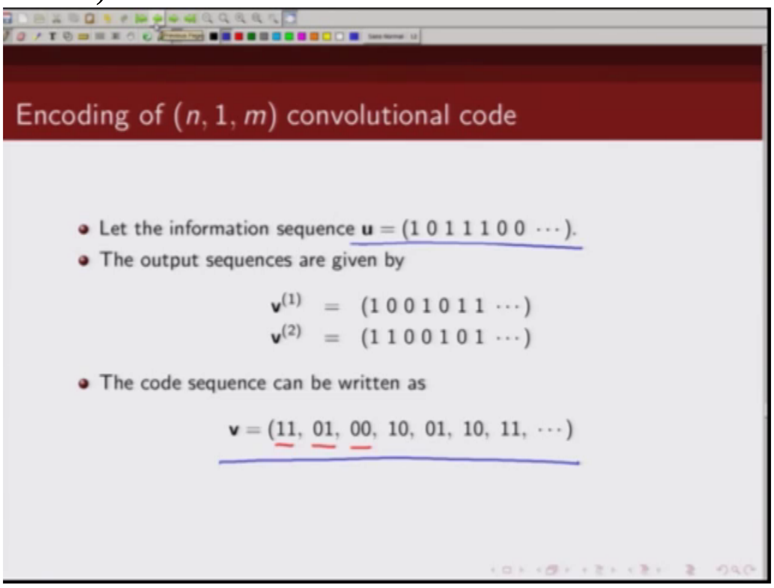

our coded sequence that we calculated in the last time. Coded sequence corresponding to this information sequence was given by this. This was our coded sequence corresponding to this

(Refer Slide Time 29:57)



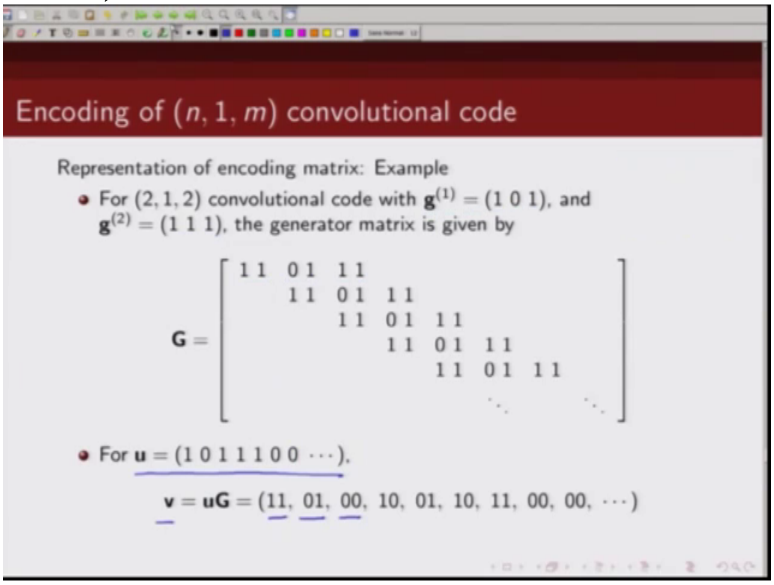

information sequence. Now let's try using this generator matrix. So if we use the generator matrix then our first input is 1 so 1 times g 0, that is this. Then next is input is zero so 1 times this plus zero times 1 1 that will be 0 1. Next 1 0 1 so 1 times 1 1, 0 times this, 1 times this so that is 0 0. So we can see basically we are getting the same output sequence. We can verify 1 1 0 1 0 0, 1 1 0 1 0 0 so we are getting the same output sequence

(Refer Slide Time 30:39)



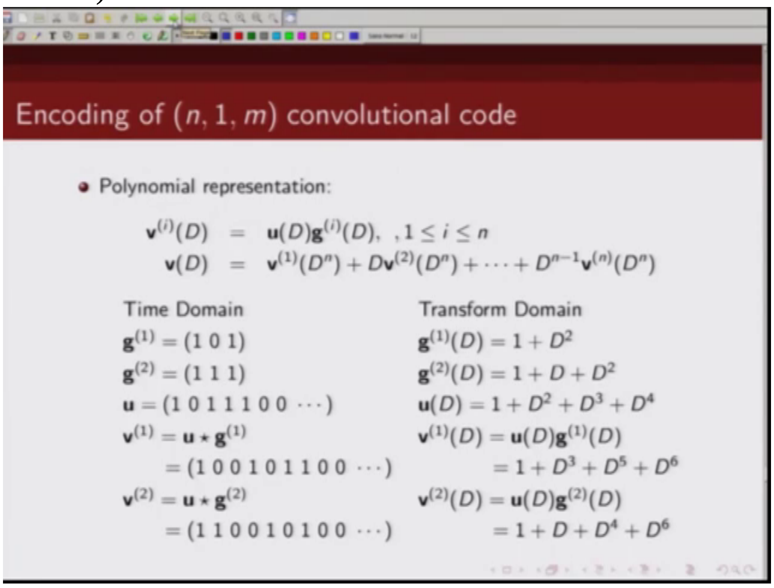

as before.

(Refer Slide Time 30:45)



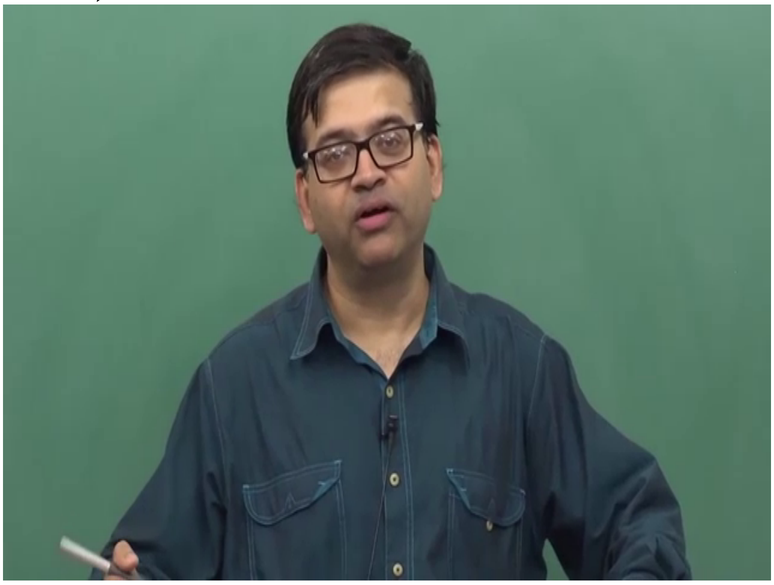

Now we are going to

(Refer Slide Time 30:49)



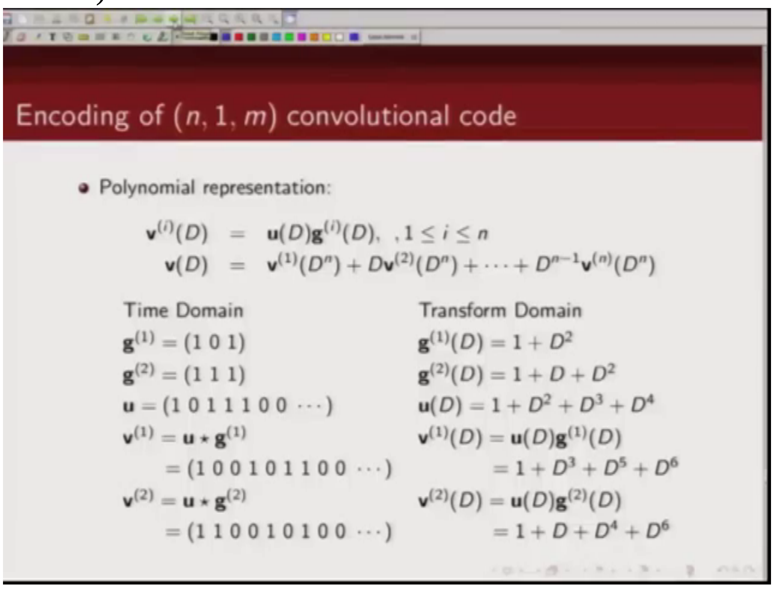

give you polynomial representation of

(Refer Slide Time 30:53)



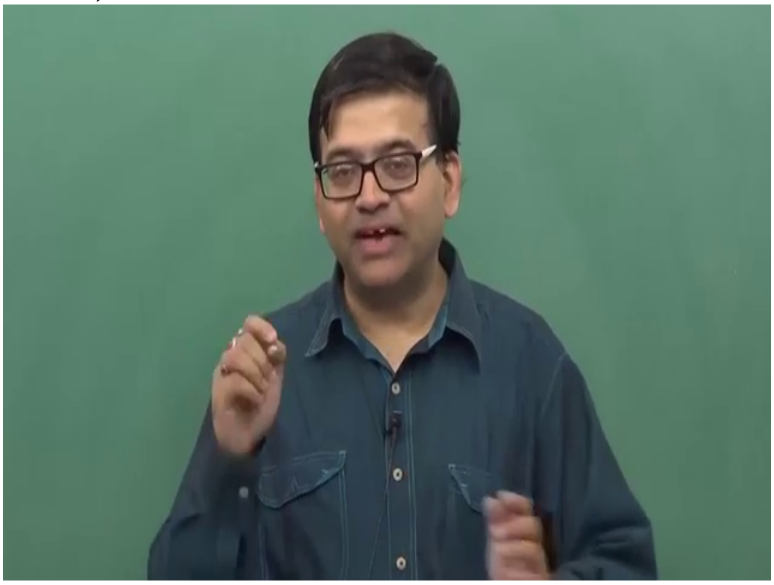
this generator sequence which is very convenient in case of convolutional codes. So I am introducing a delay operator

(Refer Slide Time 31:02)



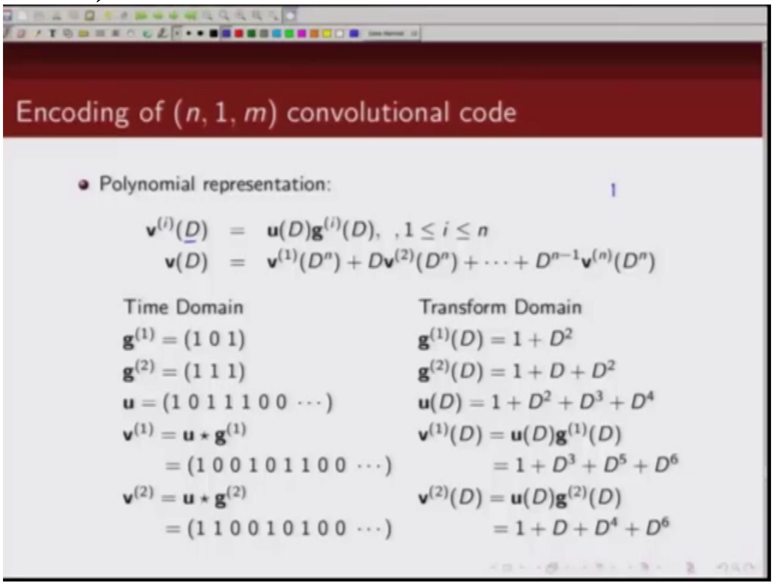

D. So if you have 1 memory element delay that will be D.

(Refer Slide Time 31:07)



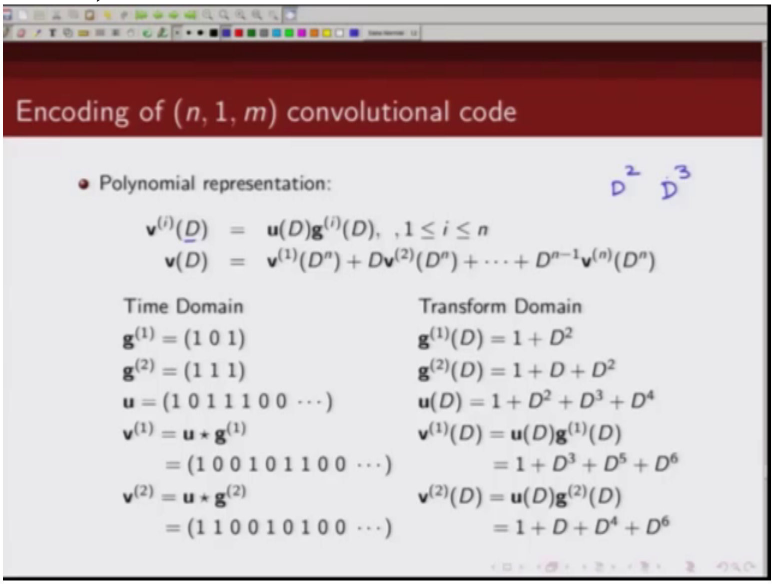
If you have delay of 2,

(Refer Slide Time 31:10)



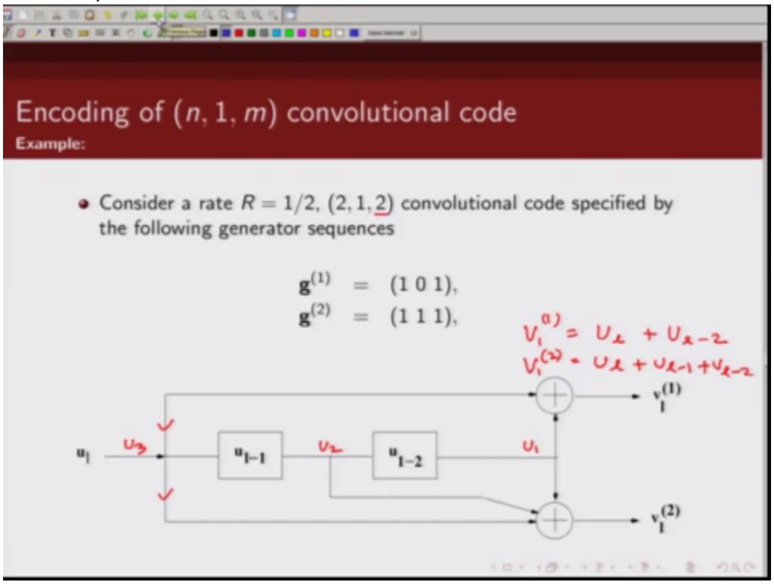

it will be D square. If you have delay of 3, it will be D cube. So the exponent of D is going to specify how much delay, Ok. so what I am going to show you

(Refer Slide Time 31:23)



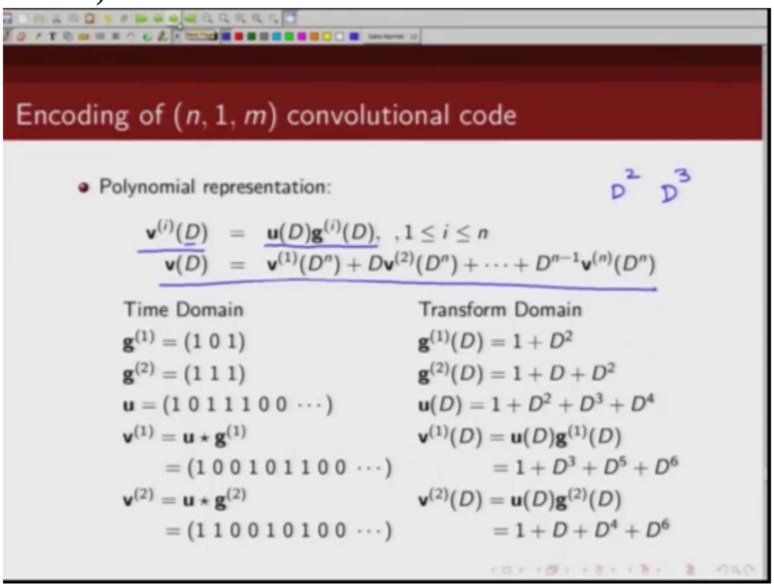

is that I can write my output sequence in this polynomial notation as u times D into g i times D. So every output code sequence can be represented as product of this information sequence using this delay operator multiplied by this generator sequence in this delay operator framework and the overall code sequence when we have rate 1 by n code can be given by this expression. So let's first try to write each of these terms in terms of delay operator polynomial representation and then we will show that this time domain representation where we were computing the output during convolution, discrete convolution can be similarly obtained using just this operation in the delay domain which we are calling transform domain operation. So we will take the same example that we were considering. So I will just go back and show you again the convolutional encoder. So this is the convolutional encoder that we are considering.

(Refer Slide Time 32:31)



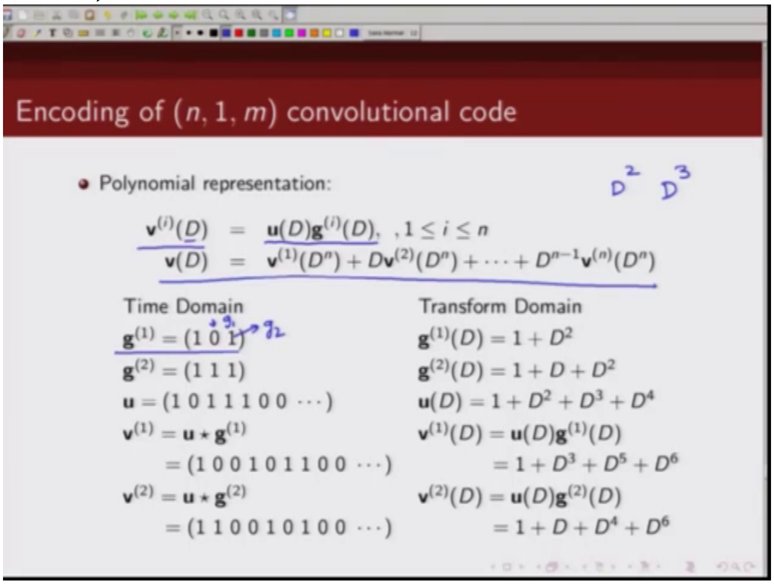

We have 1 input, we have 2 outputs. Output depends on past 2 inputs, so basically memory order is 2. g 1 is given by this; g 2 is given by this. These are my output, v 1 1, v 1 2 these are my output sequences, Ok. So let's look at

(Refer Slide Time 32:51)



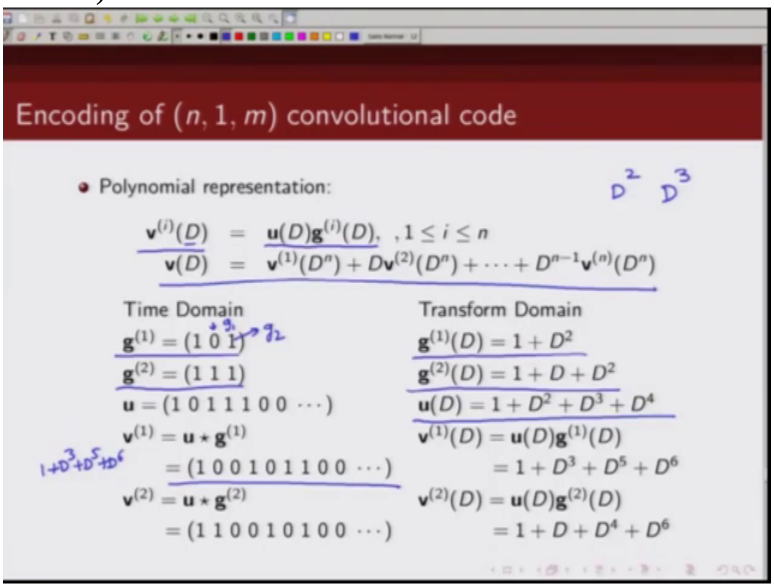

this. So g 1 is 1 0 1. Now what does this 1 corresponds to? 1 corresponds to this connection g 0 which is linking my input u l, so that would be u l without any delay. So that would be 1. What was this? This corresponds to g 1, that is input delayed by 1. So this will be represented using D so D times 0 will be 0 and this will be, this will correspond to g 2 basically and

(Refer Slide Time 33:29)



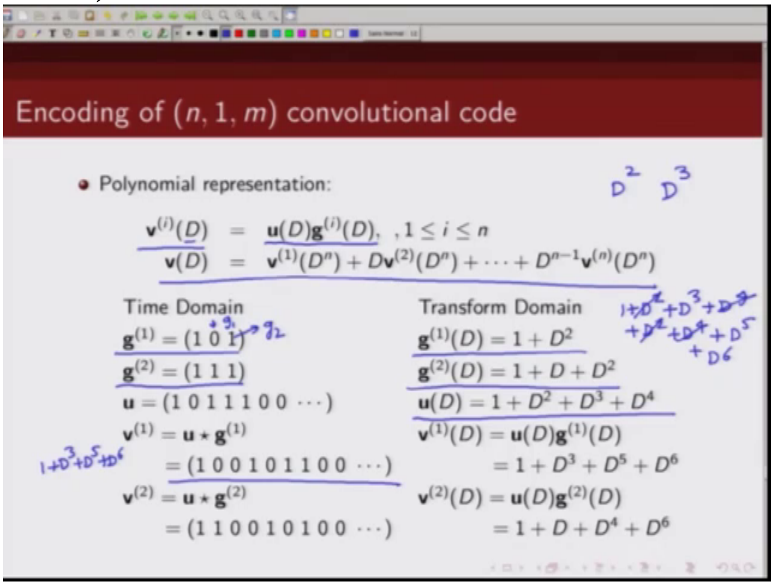

this is delay of 2, so this will be represented using D square. So this g 1 in this transformed domain using this delay operator can be written as 1 plus D square. Similarly this g 2 which is 1 1 1 can be written as 1 plus D plus D square. So this is my g 2 of D. Now information sequence also I can write in this delay notation, it is u 0, u 1, u 2, u 3. So this is information sequence I am getting at this time, this after 1 delay element, 2, 3, 4 so this would be 1 plus D square plus D cube plus D four and that's basically my information sequence. Now the discrete convolution of information sequence is g 1 is basically given by this and this if I write in uh delay operator form, will be what, 1 plus D D square D cube, D cube plus D four D five plus D six. And what is u D? u D is given by this;

(Refer Slide Time 34:52)



g 1 D is given by this. So let's multiply these 2. So what do we get? So if you multiply u D by g 1 of D so one times this will be 1 plus D square plus D cube plus D four plus D square times 1 is D square, this will be D four, this will be D five and this will be D six. D square plus D square is zero, D four plus D four is zero, so what we are left with, 1 plus D cube plus D five

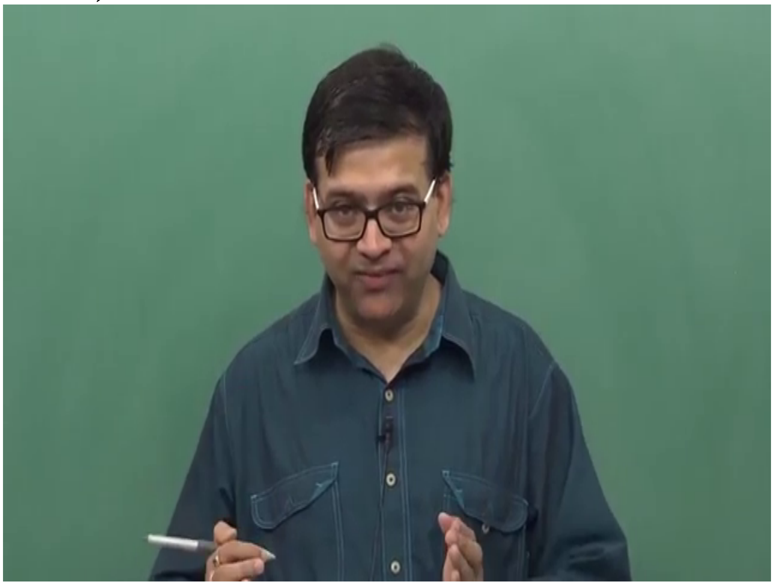
(Refer Slide Time 35:29)




plus D six. This is precisely what you get here, Ok. So you can see basically these 2 representations is equivalent. Similarly we can write u 2 which is given by this and you can verify for yourself that u 2 D is given by this.

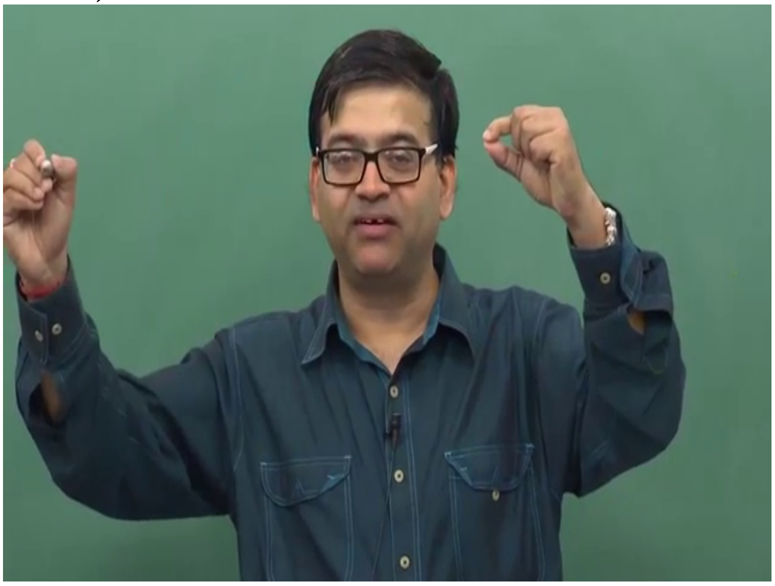
Now once you have these individual sequence,

(Refer Slide Time 35:53)



how do you write the overall output sequence? So note that for 1 input sequence you are getting n outputs, Ok. So is taken care by this. So if v i D is going to give me output sequence corresponding to each of these output, n output outputs, now I can combine n outputs in this particular fashion. So I take the first output, note that I have made here D to the power n because if it is a rate 1 by n code the first output will appear after every n bits,

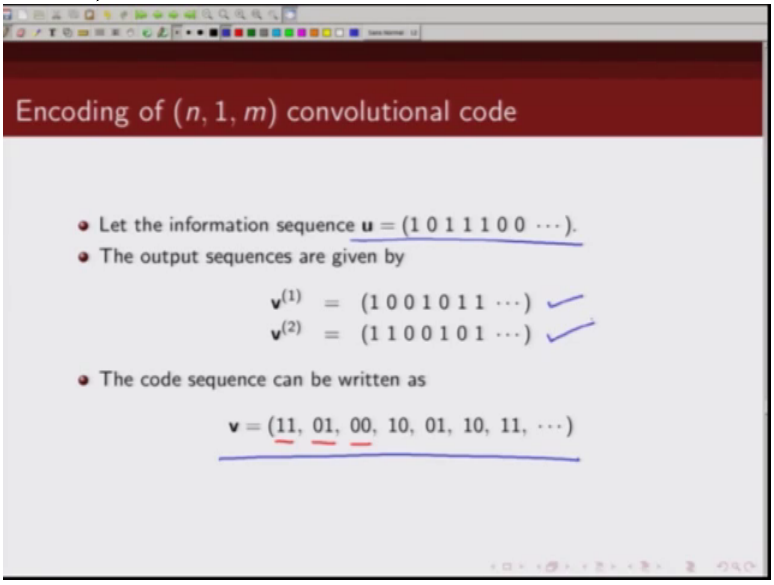
(Refer Slide Time 36:33)



the first bit is from the first coded sequence. Then after n bits, it will again repeat. It will come, meaning. So that's why I have made it D n.

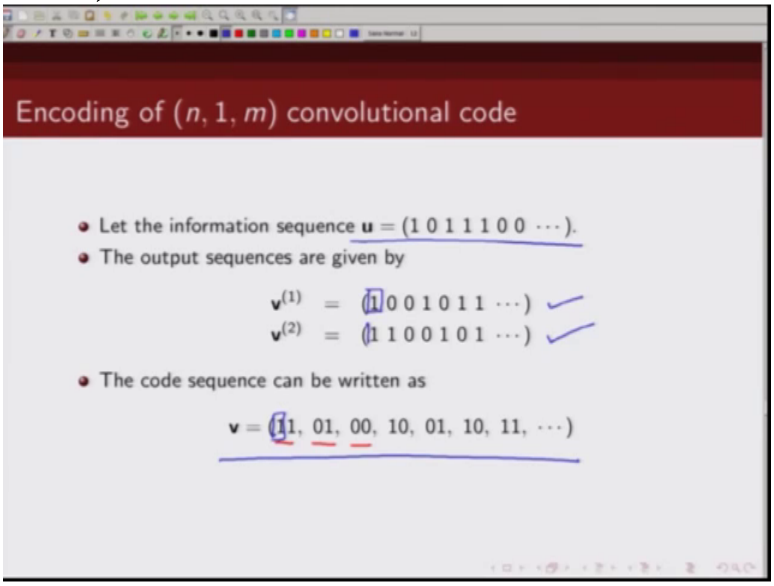
Now how do I combine these n sequences? So note, this is the output, v 1 D n is the output from the first coded sequence. This is the output from the second coded sequence. That's delayed by 1. The output from third sequence is delayed by D square. Similarly the output from the n th sequence will be delayed by n minus 1. Go back here. These are the 2 individual outputs. How are we getting

(Refer Slide Time 37:15)



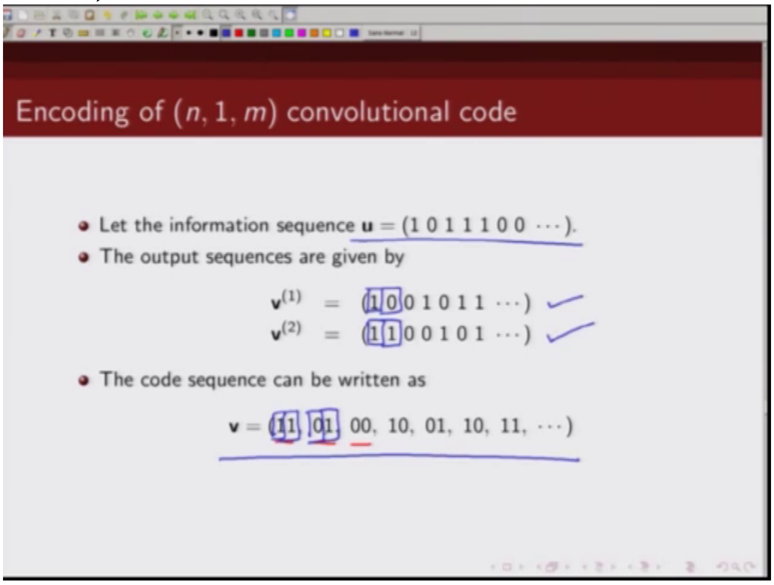

the final output? So note here. I am taking first from here. That is 1; second bit I am taking from here,

(Refer Slide Time 37:24)



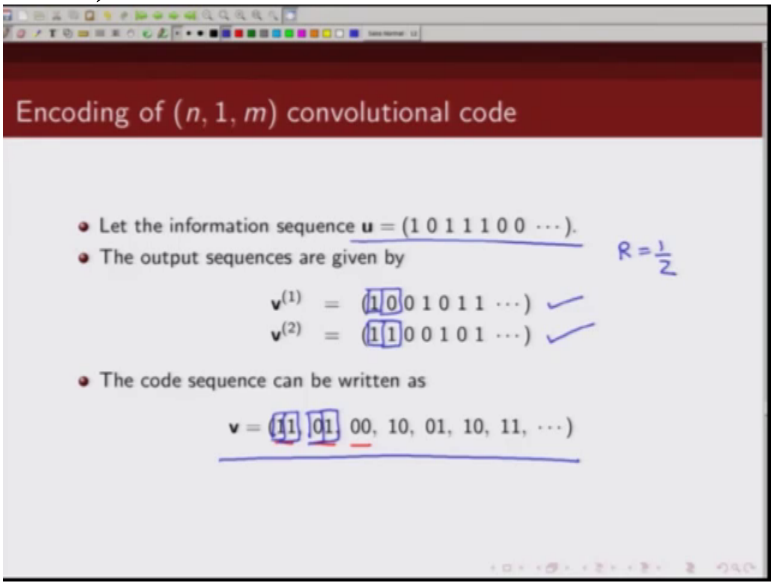

that is this. The third bit is this, which is this. Fourth bit is this which is this, so what

(Refer Slide Time 37:33)



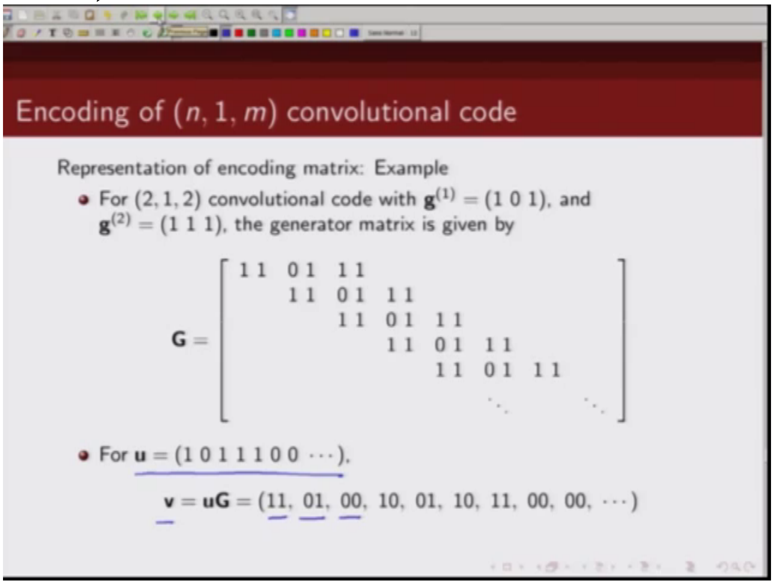

am I doing? In this case rate was one half so after every, you can see

(Refer Slide Time 37:40)

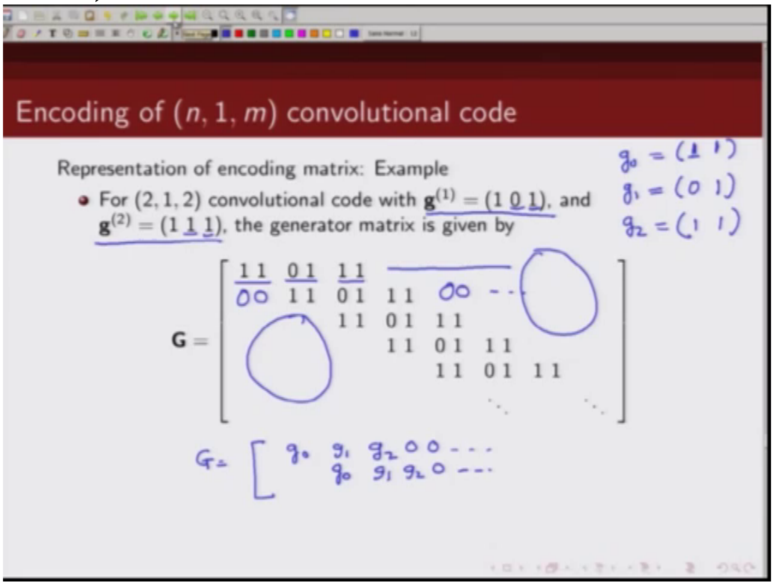



the output; every second bit is coming from this. So this is my 1 which is appearing here. This is my zero which is this. This is my zero which is this. This is my 1 which is appearing this. So note this is appearing every second bit and that's why what we did was, when we combined we made it, each of these coded bit, we made it D raised to power n. Next

(Refer Slide Time 38:08)

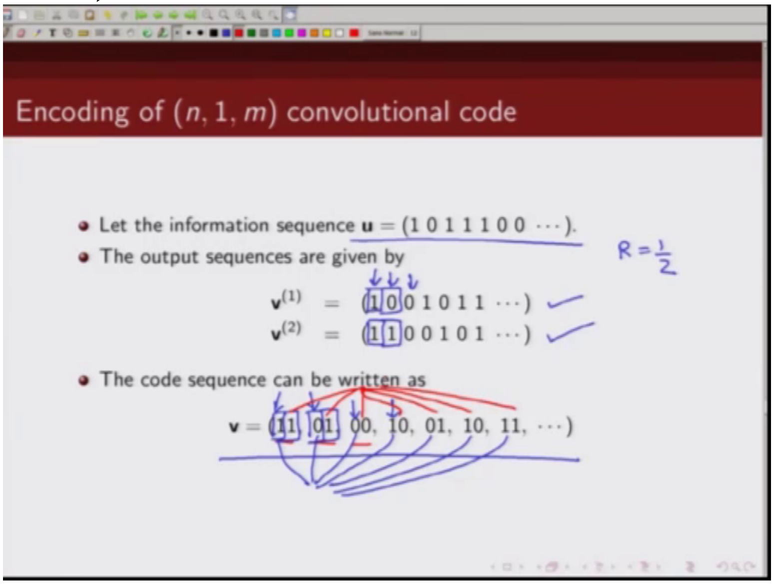



(Refer Slide Time 38:11)



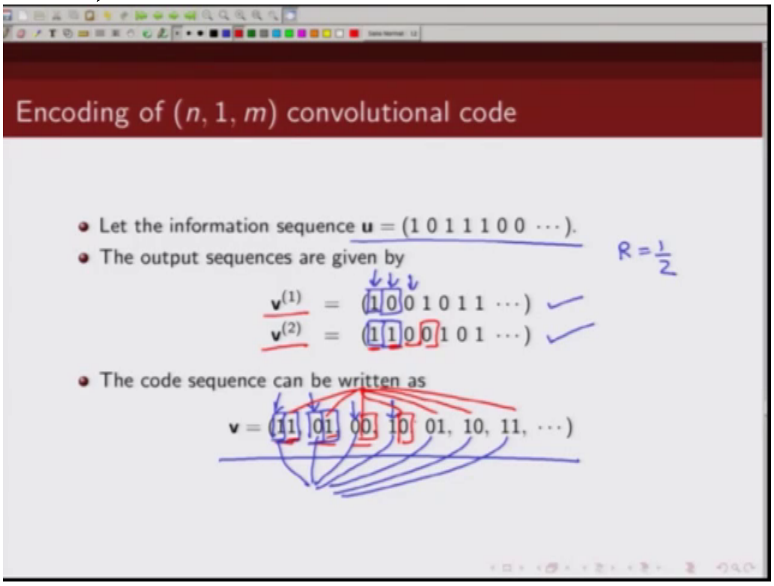

if you look here, the first coded sequence is this one. This is the output from the first coded sequence. And what is the output from second sequence, which is this one. So what are you doing?

(Refer Slide Time 38:30)



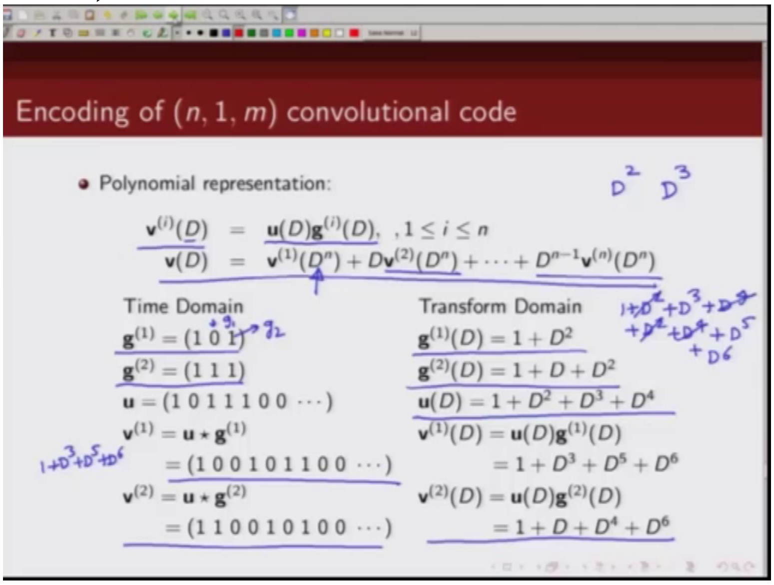

When you are combining these output sequence which is v 1 and v 2, so you are taking v 1 like as it is, only thing is it is spread out after every second bit and v 2 is delayed by 1 and it is also spread out, this is 1, this 1 is appearing here, this zero is appearing here, this zero is appearing here. So every second bit is also

(Refer Slide Time 38:56)



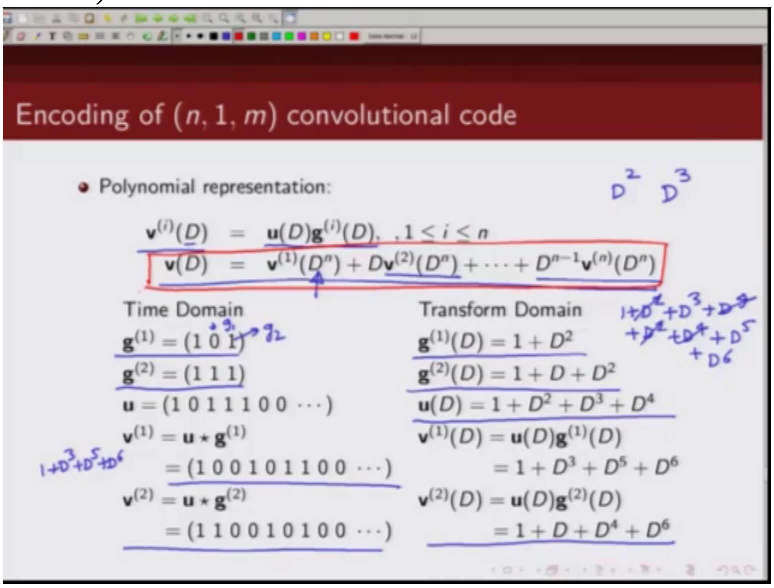

from this encoded sequence and note that this is delayed by one corresponding to v 1. So that's what we are doing here.

(Refer Slide Time 39:06)



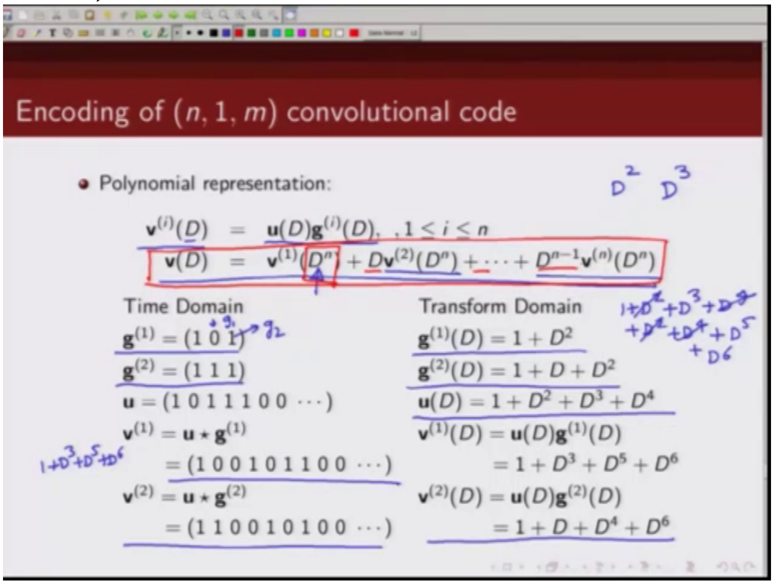

If you combine this, consider this combined output sequence there are n coded sequence, v 1, v 2, v 3, v n. Now first sequence we just take v 1 D to power n, second is v 2 D to power n, third is v 3 D to power n and then each one of them are delayed by 1 1. So this no delay, this is delay of 1, delay of 2, this is delay of n minus 1. So overall code sequence will be given by this expression. So I hope I made it clear

(Refer Slide Time 39:41)



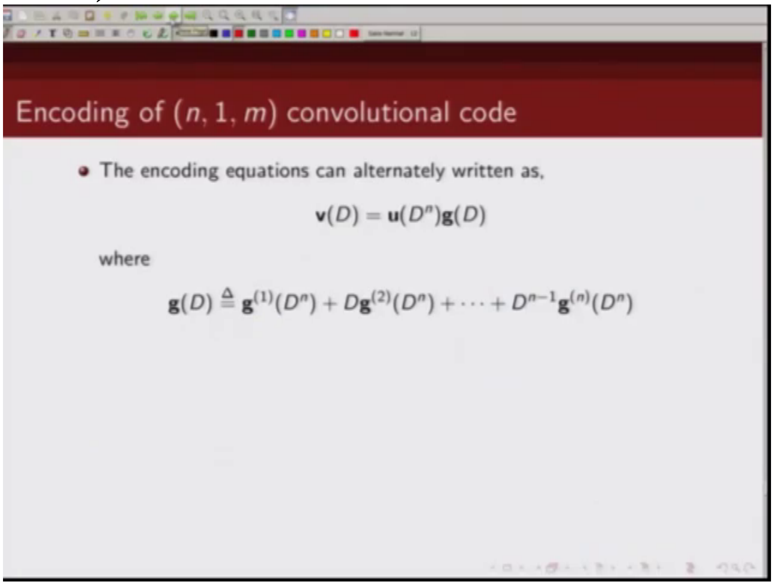

why this is D raised to power n and why each of the parity bits are delayed by 1 power like D ,D square, D cube

(Refer Slide Time 39:51)



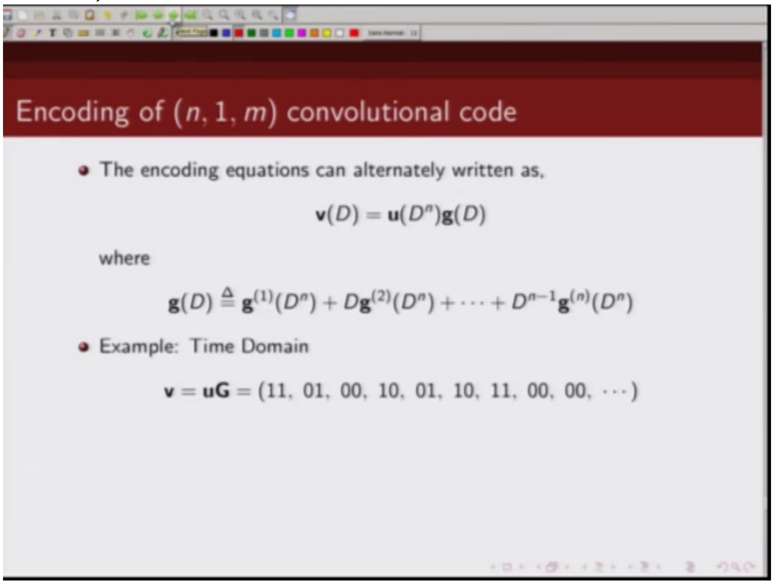

D n minus 1. So following this basically we can

(Refer Slide Time 39:55)



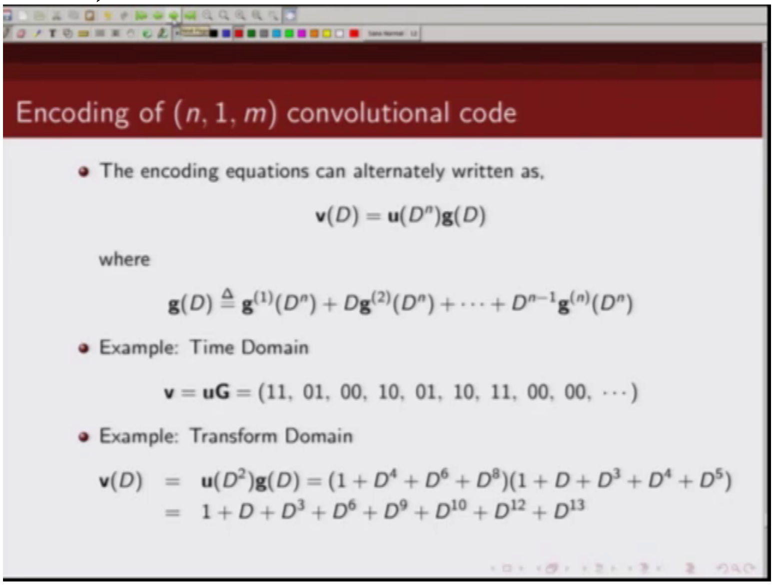

also write our encoding sequence in this particular form where output sequence is given by u D n times g D where g D is, this is generator sequence for the first coded sequence, the generator sequence for the second delayed by 1, generated sequence of the third delayed by 2, generated sequence of the n delayed by D n. So the overall encoding sequence can be equivalently written like this.

(Refer Slide Time 40:26)



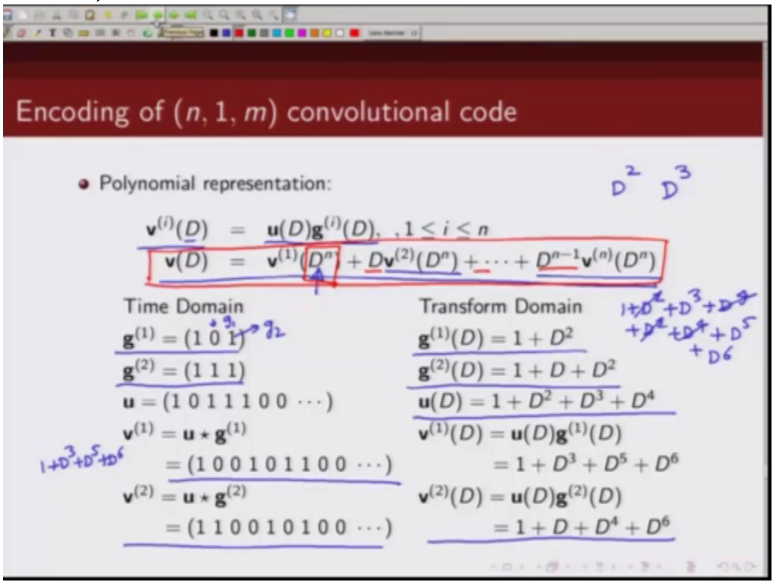

And we can again go back to the same example. Our output sequence in the time domain was given by this. And if we follow the same procedure,

(Refer Slide Time 40:36)



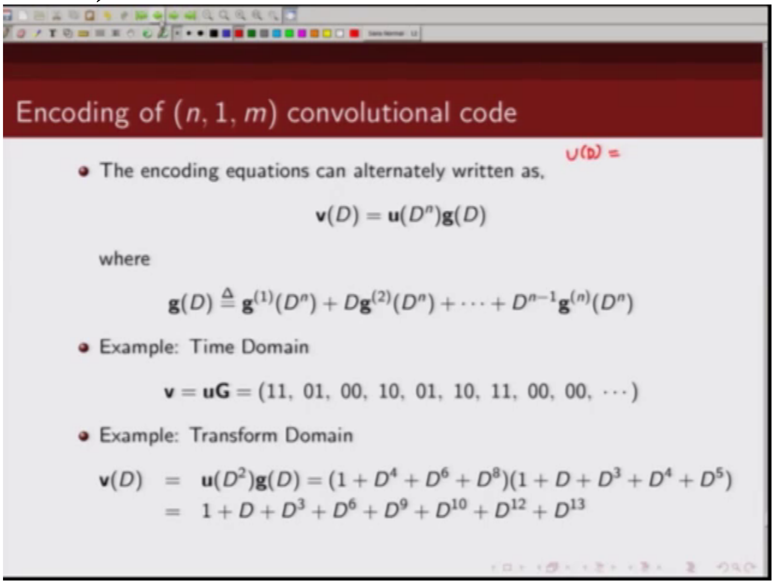

v of D should be u of D square times g D where g D is g 1 D square plus g D 2 D square. So u D D square is, what is u D? What is u D? Go back to the example. u D is 1 plus D square plus D cube plus D four.

(Refer Slide Time 41:01)



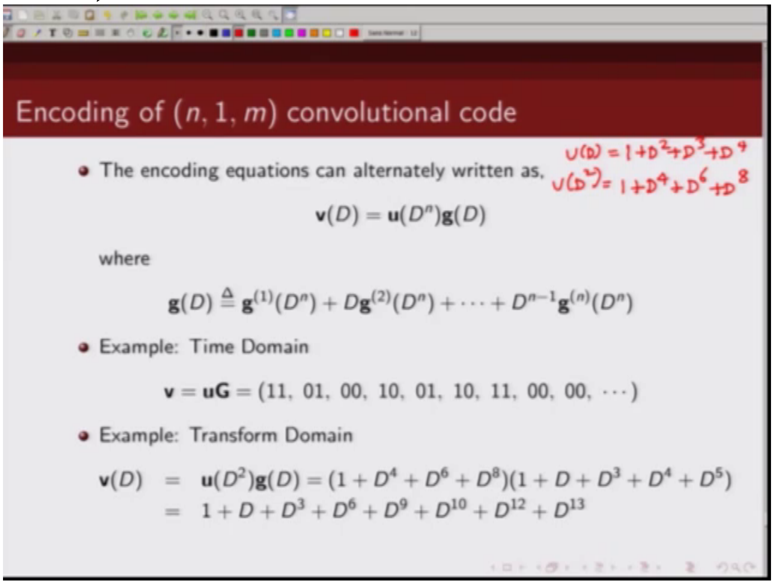

So this is,

(Refer Slide Time 41:06)



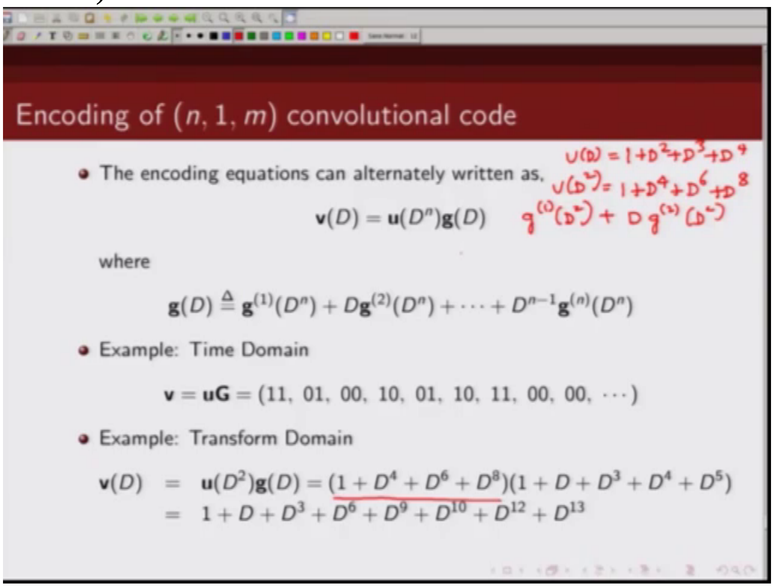

u D is 1 plus D square plus D cube plus D four. So u D square will be 1 plus D four plus D six plus D eight, so that's what we have written

(Refer Slide Time 41:22)



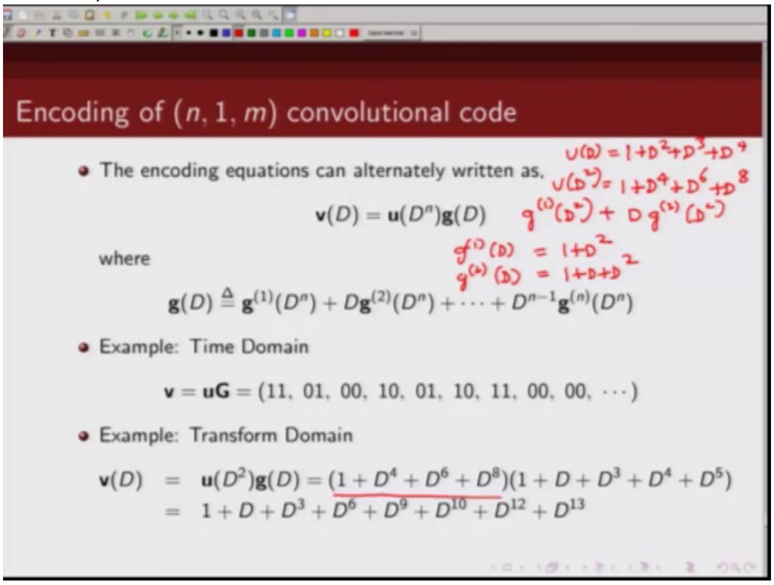

here, Ok and what is g D? g D is g of 1 D square plus D times g two of D square and what is

(Refer Slide Time 41:40)



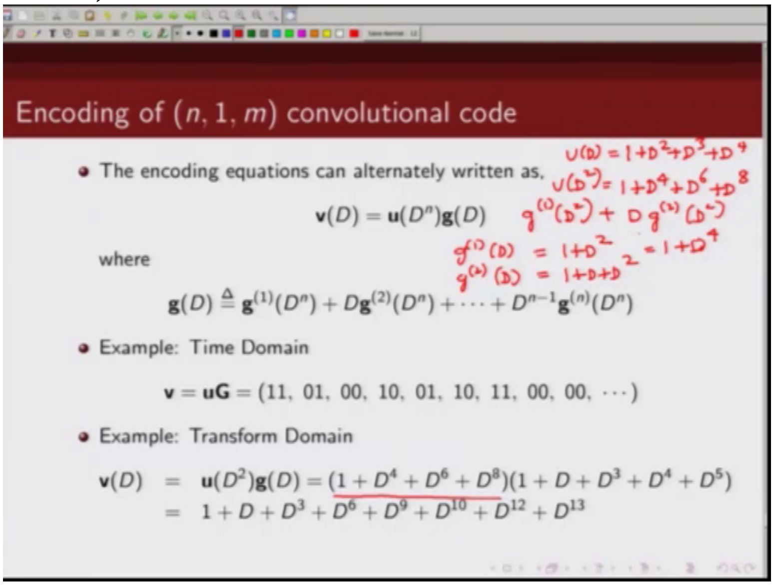

g 1 D and g 2 D? g 1 D is 1 plus D square and this was 1 plus D plus D square. So g 1 D square will be 1 plus D four

(Refer Slide Time 41:53)



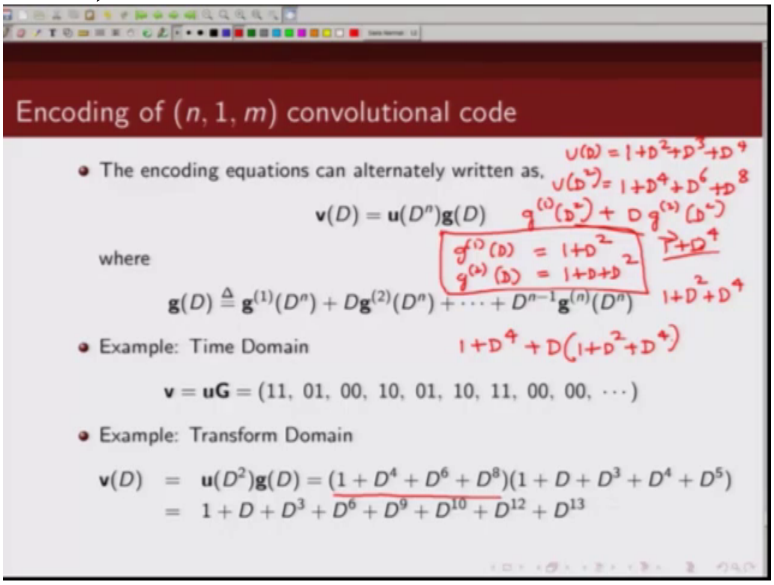

and this will be, so this will be 1 plus D four, 1 plus D four and g 2 D will be, g 2 D square, this is,

(Refer Slide Time 42:08)



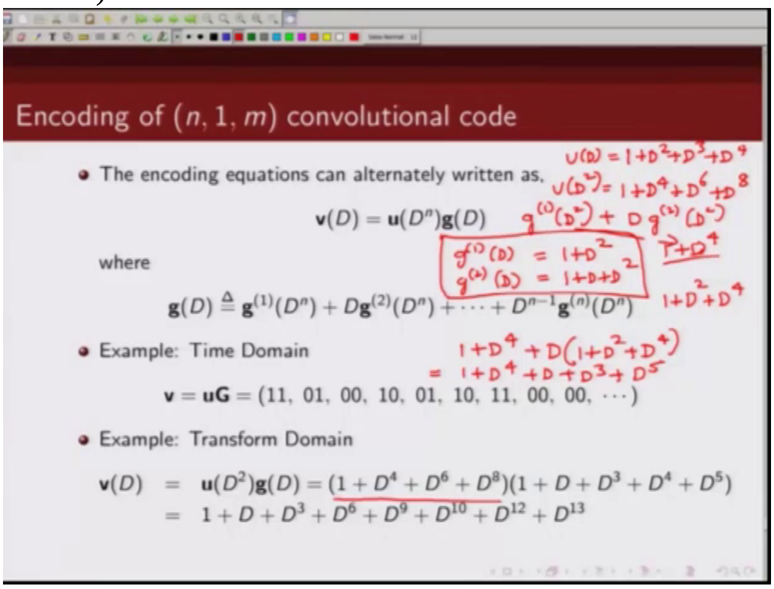

so this, this term is given by this. I hope this is clear. So g 1 D is given by this. What we are interested is g 1 D square, so g 1 D square will be given by this expression and we are interested in g 2 D square. So this will be given by 1 plus D square plus D four. Now what is our overall g D? This is given by g 1 D square plus D times g 2 D square. So then this will be g 1 D square is 1 plus D four plus D times this, Ok. So this can

(Refer Slide Time 42:55)



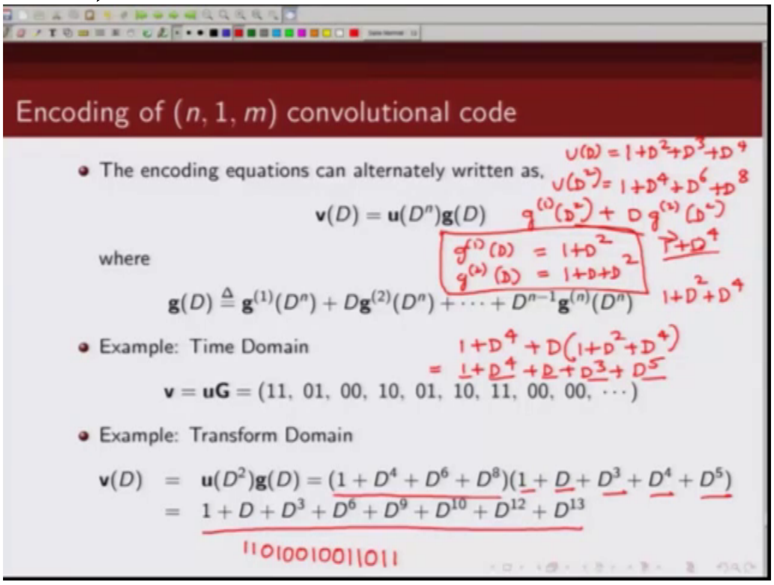

be written as 1 plus D four plus D plus D cube plus D five. So this is one plus D, this is 1,

(Refer Slide Time 43:06)




this is 1,D D, D cube D cube, D four D four, D five D five, Ok. So this is our g of D. Now if you multiply all of them, what we get is this. And we can write this, what is 1, 1 is 1, d is this, d square is 0, d cube is 1, d four is 0, d five is 0, d six is 1, d seven is 0, d eight 0, d nine 1, d ten 1, d eleven 0, d twelve 1, d thirteen 1 so this is our output sequence. Now compare with this, what we got in time domain,

(Refer Slide Time 43:50)

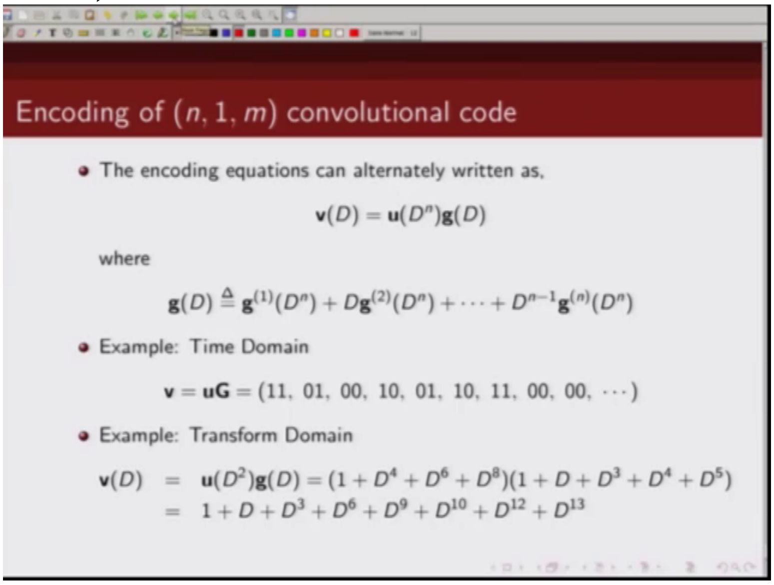



1 1, 1 1, 0 1, 0 1, 0 0, 0 0, 1 0, 1 0,0 1, 0 1.Sio you can see we are basically getting the same sequence, 1 0 1 0, 1 1 1 1 and rest are all zeroes, here also we are getting all zeroes. So the point to take is, these generator sequence that we

(Refer Slide Time 44:17)

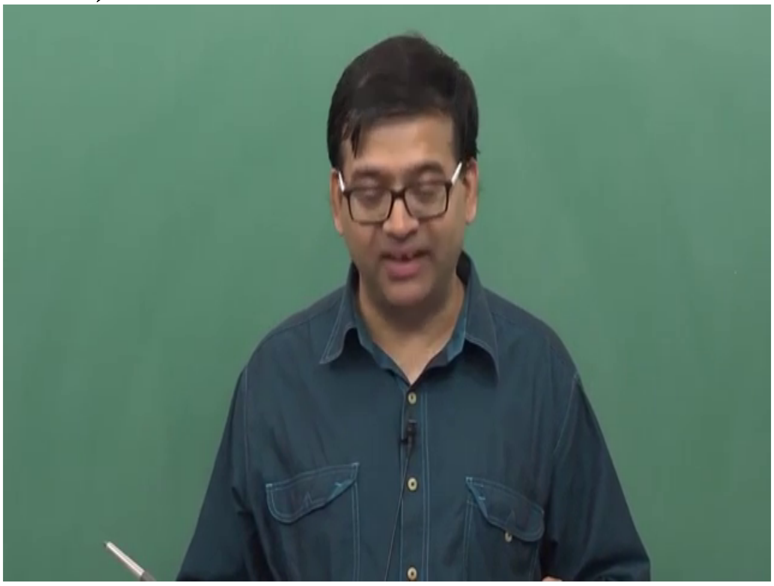
wrote using this time domain representation, we can similarly represent them using this display domain representation and it is lot more convenient to write in this particular notation because then the output sequence is just product of the input sequence and this generator sequence in this domain. So with this I am

(Refer Slide Time 44:44)



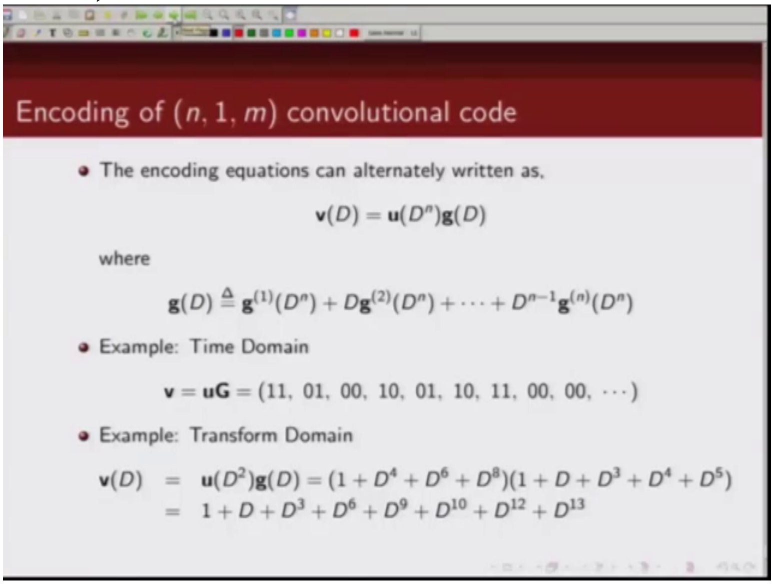

going to conclude this lecture. I just want

(Refer Slide Time 44:46)



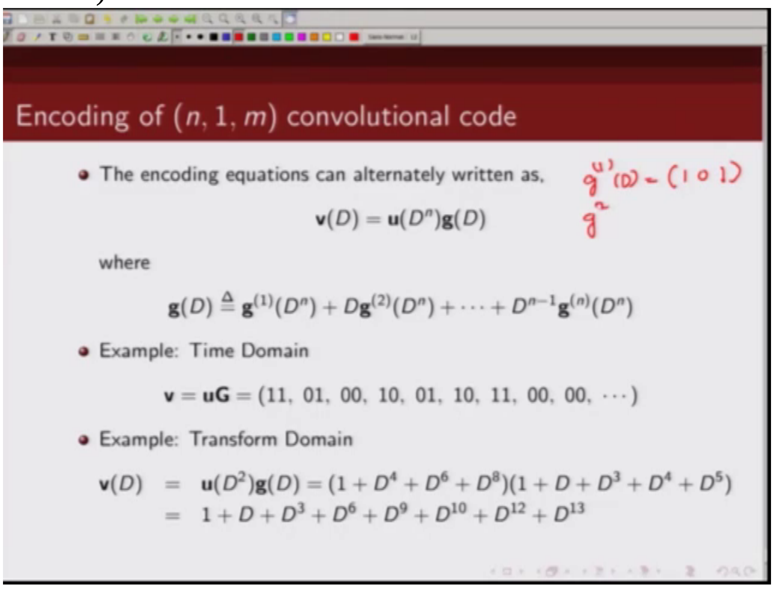
to make another point, that these generator matrix that we

(Refer Slide Time 44:50)



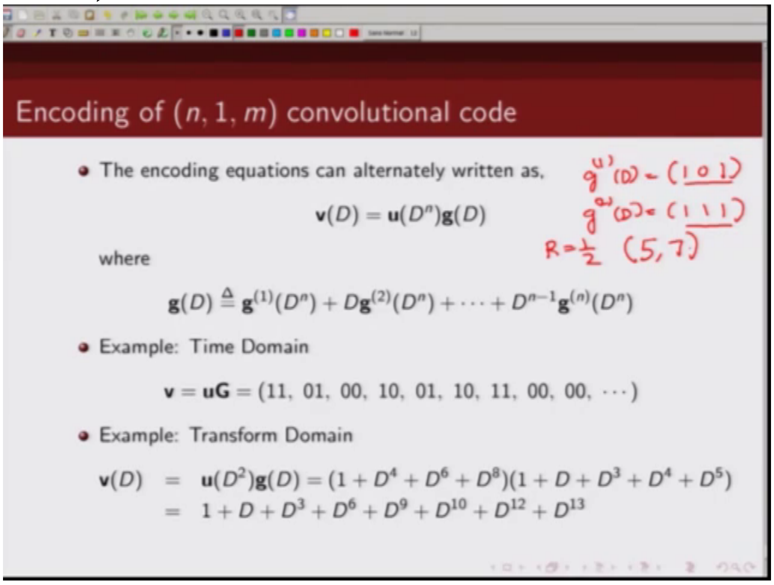

saw, for example g 1 of D which we wrote as 1 0 1 and g 2 of D

(Refer Slide Time 44:58)




which is 1 1 1, it is typically represented using octal notation. So in many books when they will describe the convolutional encoder for this, they will write it as 5 7 code because octal notation of this is 5 and octal notation of this is 7, so in many places they will say it is a rate 1 by 2 5 7 code and what it means is, they are specifying the generator sequence using this octal notation,

(Refer Slide Time 45:29)




thank you

(Refer Slide Time 45:31)