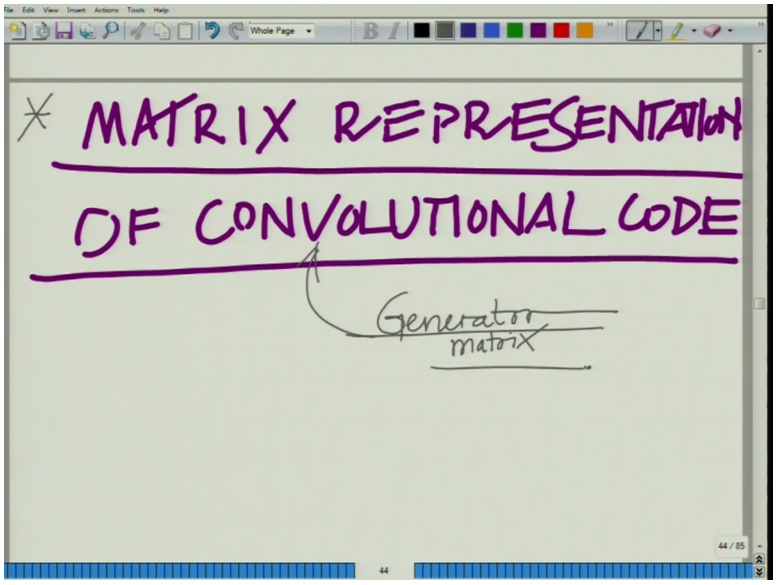
Principles of Communication Systems - Part II
Prof. Aditya K. Jagannatham
Department of Electrical Engineering
Indian Institute of Technology, Kanpur

Lecture - 51
Matrix Representation of Convolutional Code, Generator Matrix, Transform Domain Representation, Shift Register Architecture

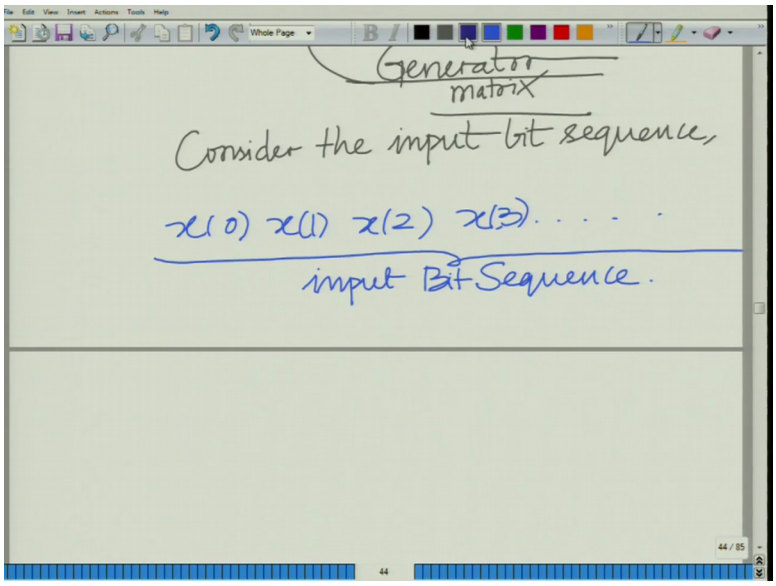
Hello. Welcome to another module in this massive open online course. We are looking at convolutional codes. Let us look at the matrix representation of a convolutional code.

(Refer Slide Time: 00:24)



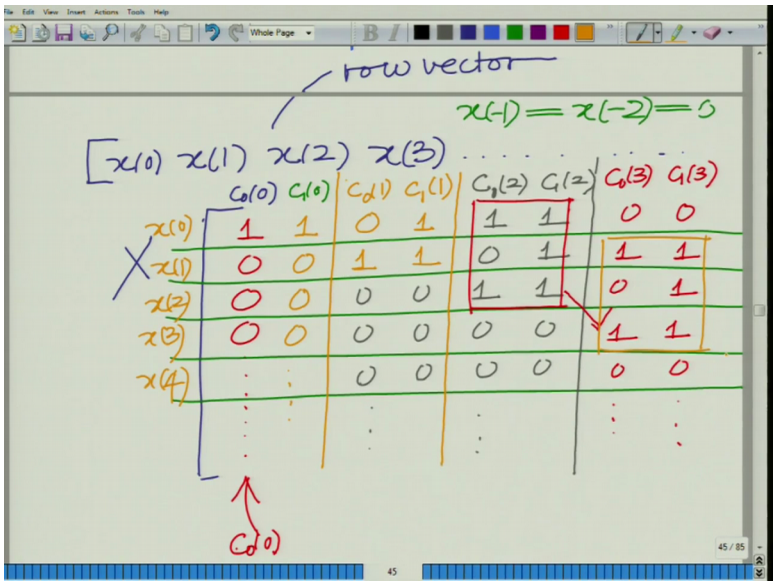
So, what we want to look at is matrix representation of a convolutional code and what we have is, we want to find the generator matrix view as a linear operation form what is known as the generator matrix for this convolutional code.

(Refer Slide Time: 01:19)



Now, consider the input bit sequence. Again as usual consider, remember for any we have to start together coded bits sequence. We have to start with the input bits sequence which let say x0 x1 x2. This is your input bit sequence.

(Refer Slide Time: 01:59)



Now, this input bits sequence let us write this as a row vector. So, I am going to write this as x0 x1. We are writing this as a row vector, this is your input bit sequence. Now, I want to multiply this by a matrix to generate the convolutional code. So, now, you can see let say this column, the first column corresponds to c0. Now, we have x0 c0.

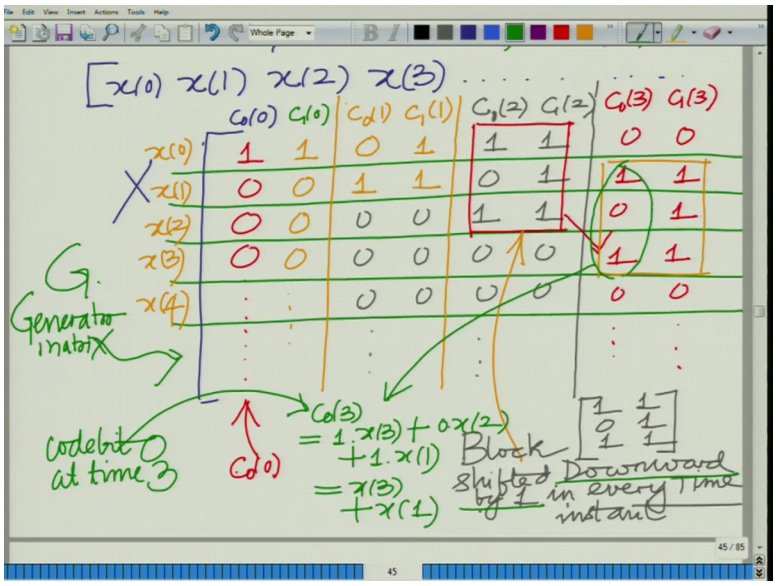
Remember c0 0 is x0 plus x of 0 minus 2, that is x of minus 2. Now, if we set x of minus 1 x of minus 2 to 0, once again similar to what we did the setting that is the previous bits before the starting of time time t equal to 0. If we set those dummy bits to 0, what we have is rather we have setting x of minus 1 equals x of minus 2 equals 0. So, what we have if c0 0 is simply x of 0 x of 0 plus x of minus 2, but x of minus 2 is 0. So, this is simply x of 0. So, this is x of 0 went to 1.

Similarly, of course x of 1 does not enter the picture. So, it is going to be multiplied by 0. x of 2 also does not enter the picture, go into multiply all the rest of. So, this column corresponds to c0 0. You can think of the rows as corresponding to the input information bits. For instance, this corresponds to x of 0 x of 1 x of 2 x of 3 and so on, ok.

Now, this corresponds to c of 0, c of 1 of 0 that is of course again I am sorry. This corresponds to first row is x of 0, second row x of 1, x of 2, x of 3, x of 4 and so on. Now, c1 of 0 is naturally x0 into 1 plus x 1 into x 1 does not into the picture. So, it is plus x 0 plus x minus 1 plus x minus 2 x minus 1 x minus 2 are 0. So, this is simply x of 0 into 1 plus x of 1 into 0 x of 2 into 0 and so on and therefore, now you have this block.

Now, similarly let us look at this for c 0 of 1 and c 1 of 1. Now, c 0 of 1 is simply x of 1 into 1. X of 0 does not enter the picture. So, 0 plus x of minus 1 x of minus 1 is 0, c 1 of 1 is x 1 into 1 plus x 0 into 1 plus x of minus 1 which is 0. Rest, do not enter the picture. So, they are zeros. So, similarly you have c 0 of 2, c 1 of 2 you can see c 0 of 2 is x of 2 plus 0 times x of 1 plus one times x of 0. C 1 of 2 is basically this is you can see this is one times x of 2 plus one times x of 1 plus one times x of 0. Rest x 3 x 4 etcetera do not enter the picture and you can see in every subsequent thing, this block will shift by 1. You can see c 0 of 3 and c 1 of 3 c 0 of 3 is x of 3 into 1 x of 2 into 0 x of 1 into 0 x of 1 into 1 plus rest all are 0 and c 1 of 3 is x of 3 into 1 plus x of 2 into 1 plus x of 1 into 1 0. Rest all are 0.

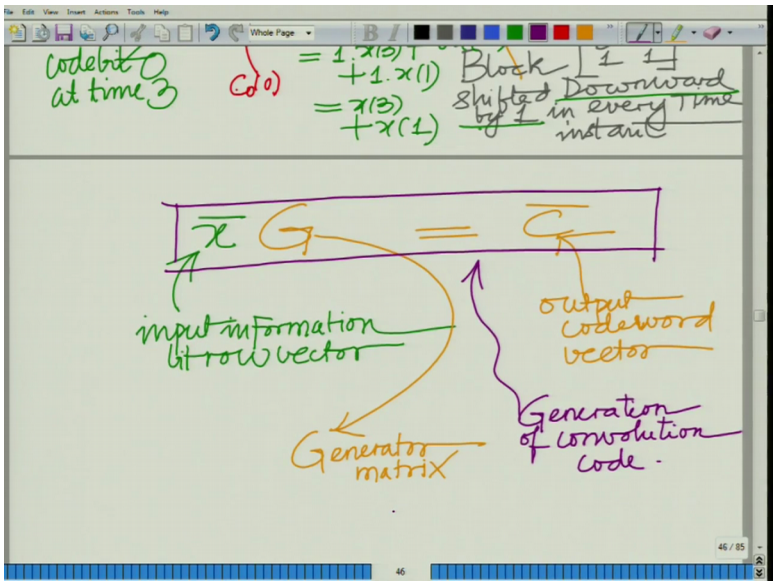
(Refer Slide Time: 06:56)



So, you can see that this block 1 0 1 1 1 1 1 is going to be shifted. So, this block is going to be shifted down by one row in every subsequent time instant. This block we are talking about this block 1 0 1 1 1 1 1 is shifted downward by one in every time instant. For instance, if you can look at this, if you can look at this C0 of 3, let me just look at this entry C0 of 3. This corresponds to now you can see C0 of 3 equals 1 times x of 3 plus 0 times x of 1 plus 1 times x of 1 which is equal to x of 3 plus x of 1. That is your code bit 0 at time 3 code bit 0 at time 3, ok.

So, this block 1 0 1 1 1 1 1 is shifted downward by one in every time instant and this matrix is known as the generator matrix for this convolutional code.

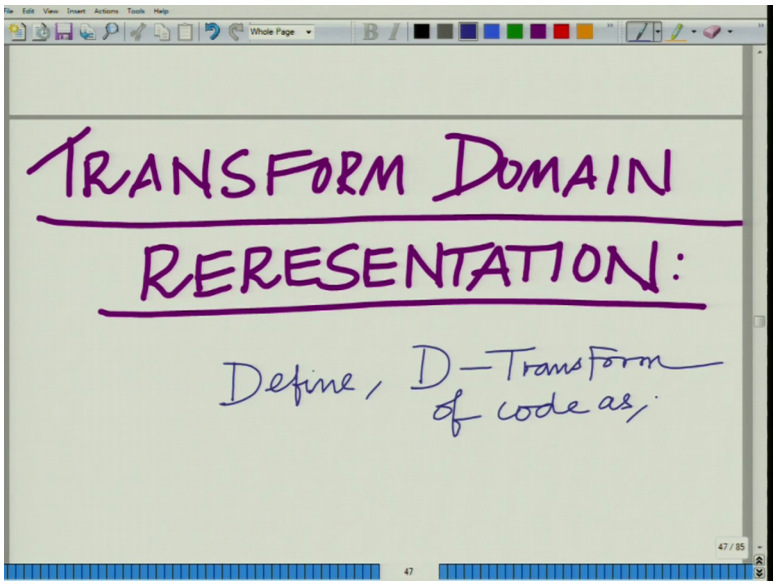
(Refer Slide Time: 08:40)



So, what I can do is, I can take the input vector x bar rho vector. This is equal to G times or basically this into the generator matrix G; this is equal to C bar which is basically your output codeword vector. So, you can represent this code generation as a linear transformation for this basically captures the generation of the convolutional code, ok.

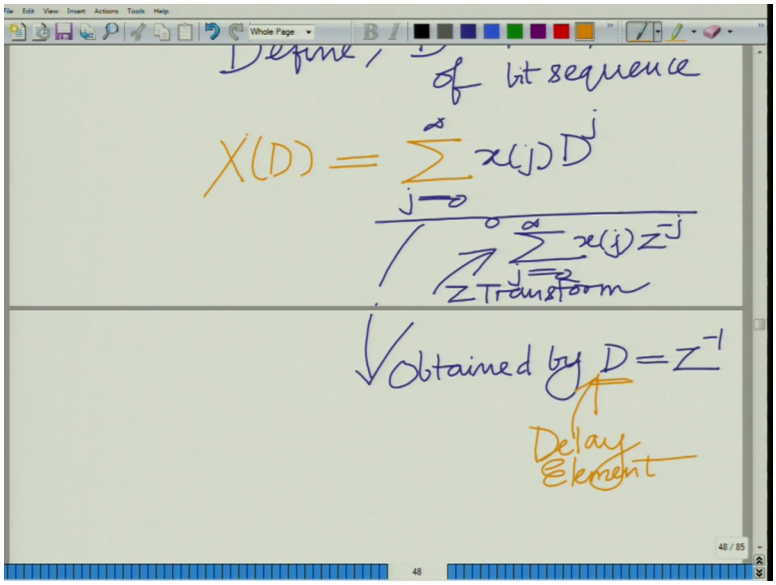
Let us look at another representation of the convolutional code. This is known as a transform domain representation or let me just rewrite it clearly.

(Refer Slide Time: 10:08)



Let us call this as the transform domain representation and we can define the D transform of the code. So, we define what is known as the D transform.

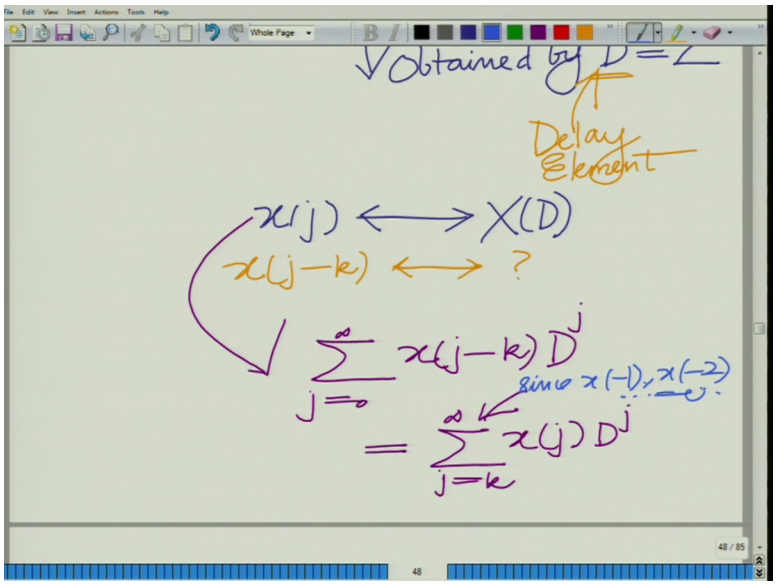
(Refer Slide Time: 11:05)



We define the D transform of the code as X of D equals this is or not the code, this is the D transform og information bit sequence. So, this is j equal to 0 to infinity x of j d to the power of j.

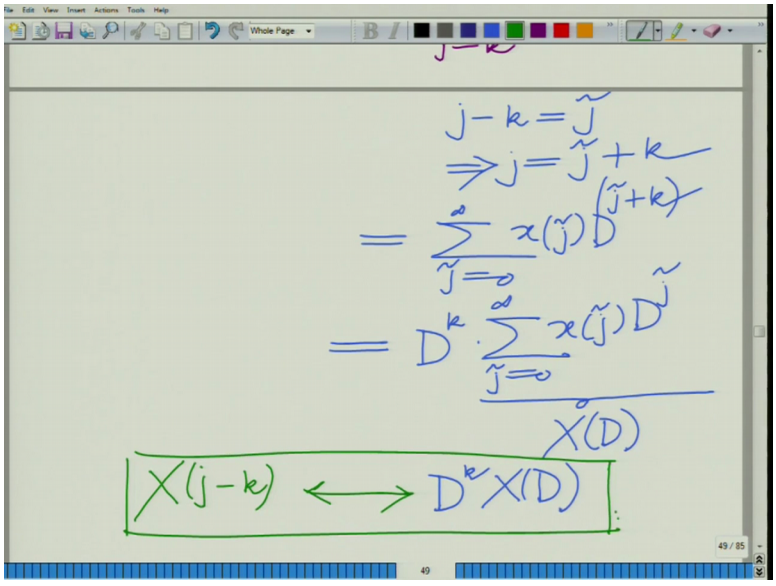
Now, observe that this is similar to the zee transform. Remember zee transform is equal to 0 to infinity x of j zee to the power of minus j. This is your zee transform, but this is not the zee transform. This is the D transform and this D transform is similarly simply obtained by setting, you can see obtained by setting D equal to. So, D basically represent remember zee inverse that simply represent the delay. So, D represents a delay element right correct. So, this quantity D simply represents a delay element or you might remember from that theory of zee transform of digital signal processing that is zee transform for a given sequence X n, the zee transform is simply summation X n zee to the power of minus n. We are replacing D by zee inverse. So, this becomes x n d to the power of n or x j d to the power of j. This is particularly convenient for our representation of an information bit sequence or for that matter also coded bit sequence, all right. So, that is known as the D transform. So, D is equal to 0 which represents the delay. So, this system D characterizes the delay and again, it goes without saying.

(Refer Slide Time: 13:14)



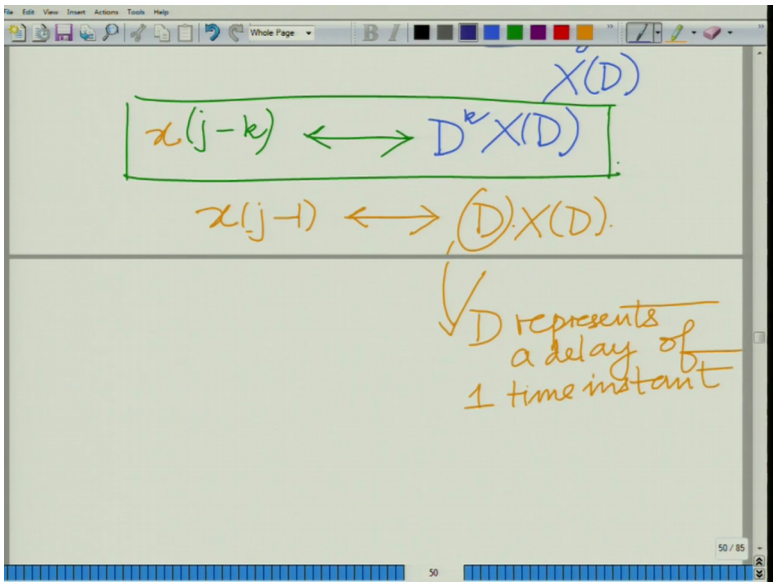
For instance, I have x of j with transform x of d x of you have x of j minus k. What is the corresponding transform? Now, of course first let us start with this x of j minus k starts from. So, you have k equal to or j equal to 0 to infinity x of j minus k d to the power of k. Now, j minus k x if x starts only at 0, j equal to 0, x of j minus k only starts at j equal to k because x of minus 1 and so on are zeros. So, this starts at j equal to k to infinity x of j d j. This is since x of minus 1 etcetera are 0.

(Refer Slide Time: 14:28)



Now, substitute j minus k equal to j tilda implies j equals j tilda plus k. So, this becomes j tilda equal to 0 to infinity x of, well sorry this should be x of j minus k j tilda d to the power of j tilda plus k and this you can clearly see is t to the power of k j tilda equal to zero to infinity x of j tilda into d raise to j tilda and this you can clearly see is x of d. So, this is simply d to the power of k into x of d. So, the transform of x j minus k is d to the power of k.

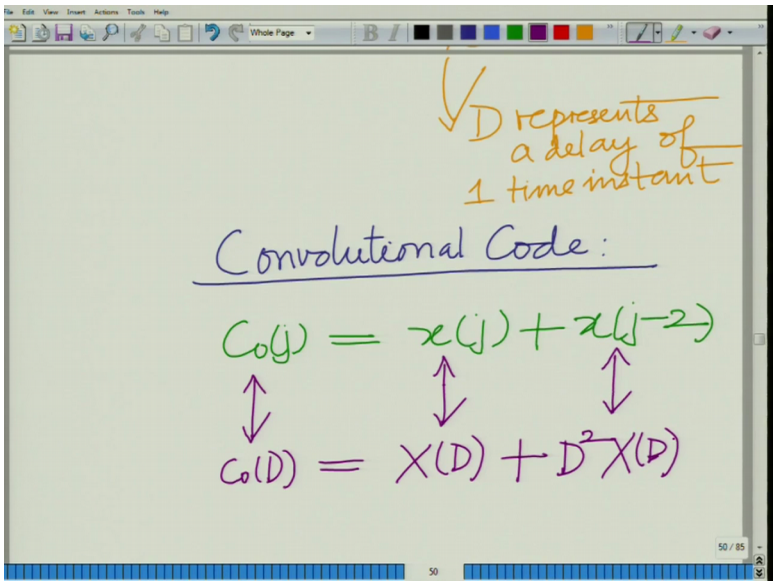
(Refer Slide Time: 15:52)



Now, in particular if you look at this x of j minus 1, that is in particular if you look at x of j minus 1, that will have a transform d time x of d and therefore, this d represents a delay that is what you can say this d represents a delay of one unit. That is what you can see d represents a delay of one time instant d represents a delay of one time instant, ok.

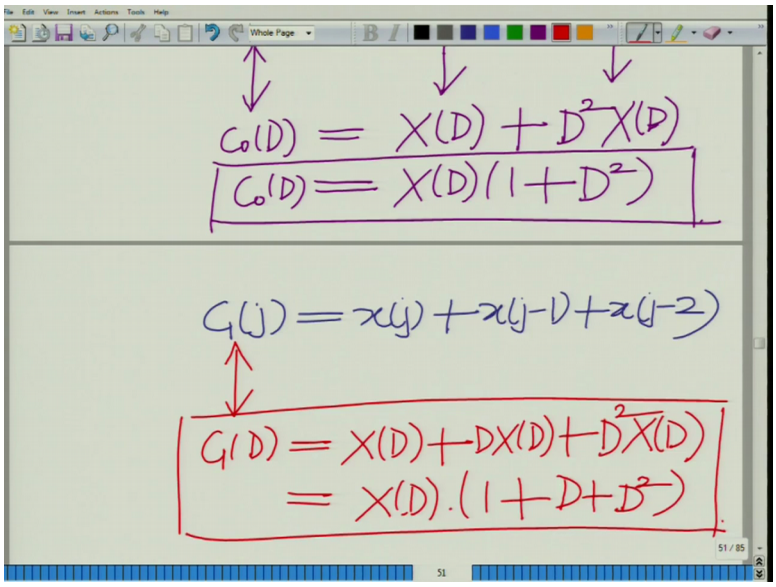
Now, let us go back to our convolutional code or let us now go back with this knowledge of Zee D transform. Let us now go back to our convolutional code and let us try to develop D domain representation of the convolutional code.

(Refer Slide Time: 16:55)



So, let us go back to our convolutional code. Now, we go back to our convolutional code and what you see is C naught of j is x of j plus x of j minus 2. If you take the D transform, what you are going to have is C naught of D is equal to x of d, D transform of this plus D transform of x of j minus 2. That we have already seen is d square times x of d x of j minus k is d to the power of k times x of d.

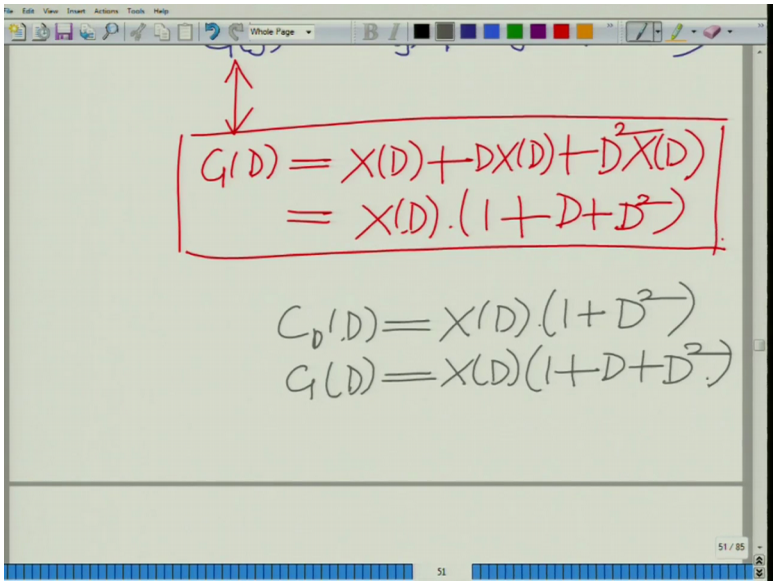
(Refer Slide Time: 17:48)



Therefore, C the D transform of C naught code bit sequence 0 is simply x of d into 1 plus d square. So, we have an elegant D domain relation to characterize the D transform of the coded bit sequence 0.

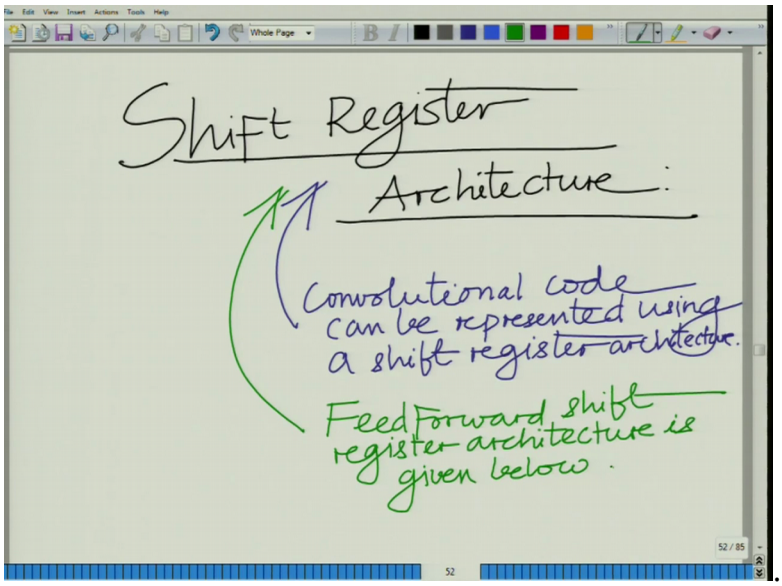
Similarly, C1 of j is equal to x of j plus x of j minus 1 plus x of j minus 2. If you take once again the D transform C1 of d equals x of d plus d times x x of j minus 1 as D transform d times x of d plus x of d minus j minus 2 d square times x of d. This is equal to x of d into 1 plus d plus d square. So, this is the relation that we have.

(Refer Slide Time: 19:02)



So, finally, to summarize this, we can write in D transform domains C0 of d equals x of d into 1 plus d square C1 of d equals x of d into 1 plus d plus d square. This can now be represented using a shift register architecture.

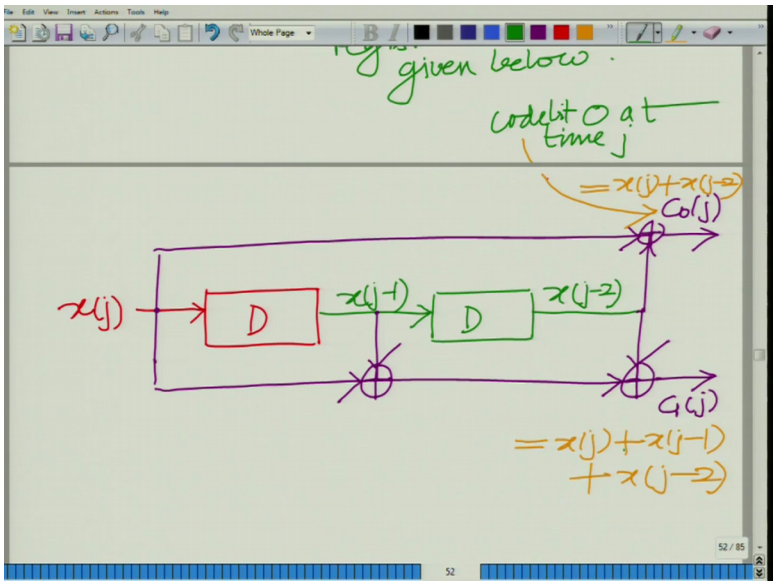
(Refer Slide Time: 19:28)



So, what we are now going to do is represent these using a shift register. We can now represent this using a shift register architecture and we are going to use a shift register architecture to represent this convolutional code which is convenient and this shift register architecture was very popular, especially in the early days of the convolutional code to practically implement in a digital communication system.

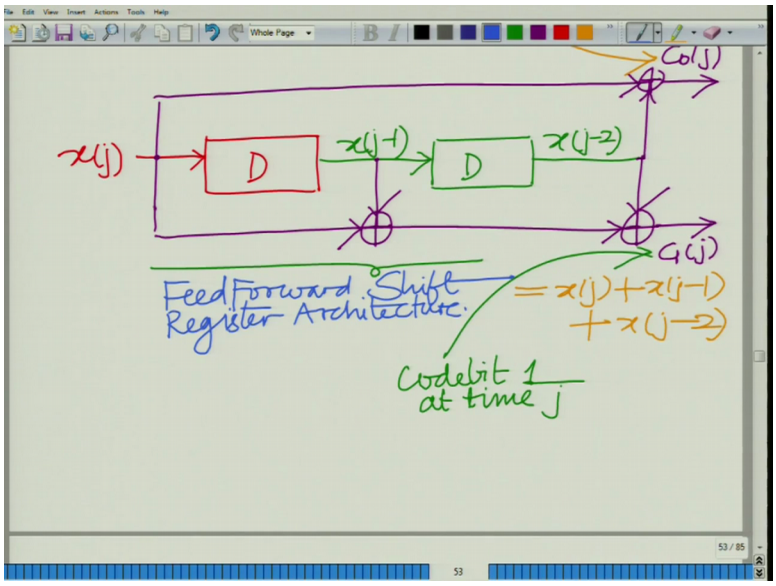
In particular, we can use a feed forward shift register architecture. I can use a feed forward. So, a convolutional code can be represented as a shift register architecture. In fact, below we are going to show a feed forward shift register architecture. A feed forward a feed forward shift register architecture is given below. So, remember we have x of j that is x of d.

(Refer Slide Time: 21:28)



So, we have x of j and it goes through a delay that gives rise to x of j minus 1 x of j minus 1 goes through another delay that gives rise to x of j minus 2, and what we can do now is basically all I have to do is take this x of j correct, x of j we have x of j and that gives me x of j plus x of j minus 2. That gives me x of j plus x of j minus 2. Let me draw this again for the sake of clarity. So, I have x of j delay that gives me x of j minus 1, delay that gives me x of j minus 2. I take x of j, I add it to x of j minus 2 and that gives this C0 of j. I take x of j, add x of j minus 1 and I also add to this x of j minus 2, sorry x of j plus x of j minus 1 plus x of j minus 2 and that gives me C1 j. So, c naught j, this is equal to x of j plus x of j minus 2 C 1 of j. This is equal to x of j plus x of j minus 1 plus x of j minus 2.

(Refer Slide Time: 24:23)



So, this is code bit 0. Again it goes without say in code bit 0 at time j and this is your code bit 1. This is your code bit 1 at time j and this is your feed forward shift register architecture. So, what we have done in this module is, basically we have looked at several things. We have first looked at a generator matrix representation of the convolutional code or a generation of convolutional code based on the generator matrix, we have constructed the generator matrix, we have looked at the D domain formulation or D domain representation of any coded bit sequence. Also, the corresponding D domain representation of the filters or basically D domain representation of a convolutional code and also, this shift register architecture for generation of the convolutional code that is the coded bit sequence given an information bit sequence, all right.

So, we will stop here and look at other aspects in the subsequent modules. that we will get the more important and interesting aspects which are basically going to be extremely important one is the state state diagram representation of the convolutional code followed by the extremely important and critical trellis diagram for the convolutional code and the decoding. Now, what we have seen so far is basically given the information bit sequence, how to generate the coded bit sequence. What we have to do now is consider the received sequence. Of course, the received sequence at the receiver might not be the same as the generated coded bit stream because as we have said the whole purpose of coding is because there can be errors over the channel.

So, now we have a received bit stream from this, recover the corresponding coded bit stream, of course recover the corresponding information bit seq. Of course, we said the information bit stream ideally we would like to recover it without any errors, but in practice that is not possible because the errors can only be made as low as possible, right. One can only target try to make the error rate as low as possible. So, trying to recover the maximal likelyhood, all right trying to recover the information bit sequence to the most accurate extent that is possible, that is going to be your aim. Look at the appropriate decoding process that is the decoding principle, the maximal likelihood principle and the corresponding viterbi algorithm which would be used to construct, to reconstruct the information bit sequence from the received bit sequence. So, we will stop here.

Thank you very much.