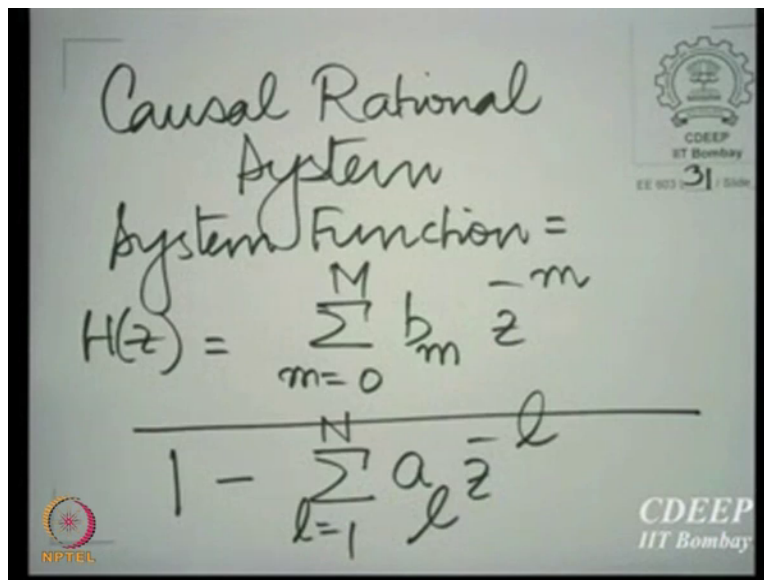


**Digital Signal Processing & Its Applications**  
**Professor Vikram M. Gadre**  
**Department of Electrical Engineering**  
**Indian Institute of Technology Bombay**  
**Lecture 31A**  
**Intro to Signal Flow Graph and Direct Forms-1 & 2**

A warm welcome to the 31 lecture on the subject of Digital Signal Processing and its Applications. We are now going to embark on a very important theme related to discrete systems after having completed a fairly extensive discussion on the design of finite impulse response filters using the window approach, a window based filter design method.

We now need to work out how we can realize systems. I in fact, we had just hinted at the approach to realisation in the past, but now we need to treat that subject with some degree of depth and detail. So, in fact, let us put before ourselves the specific problem that we need to discuss.

(Refer Slide Time: 01:28)



The image shows a handwritten slide with the following content:

Causal Rational  
System

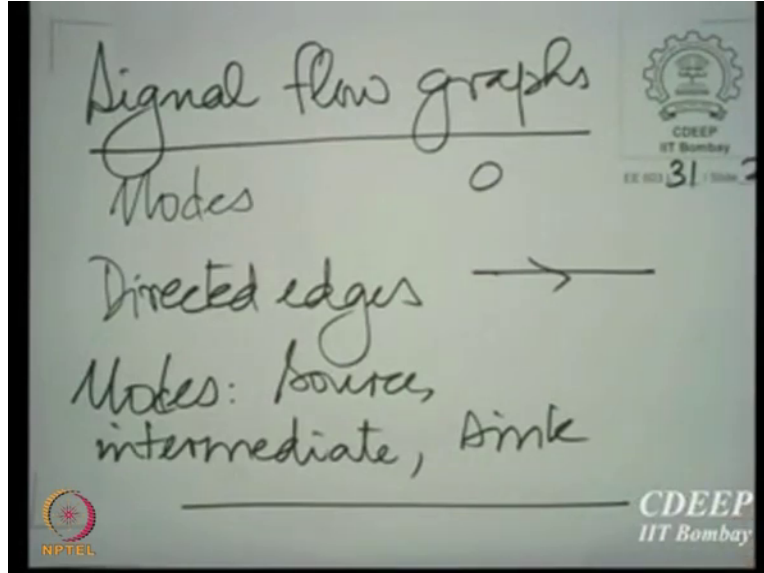
System Function =

$$H(z) = \frac{\sum_{m=0}^M b_m z^{-m}}{1 - \sum_{l=1}^N a_l z^{-l}}$$

The slide also features logos for NPTEL (bottom left), CDEEP IIT Bombay (top right), and CDEEP IIT Bombay (bottom right).

You see we have before us in realization a causal rational system and the system function takes the following form,  $H(z) = \{ \sum_{m=0}^M b_m z^{-m} \} / \{ 1 - \sum_{l=1}^N a_l z^{-l} \}$ . So, we have a numerator and a denominator polynomial and our objective is to realize this system.

(Refer Slide Time: 02:22)

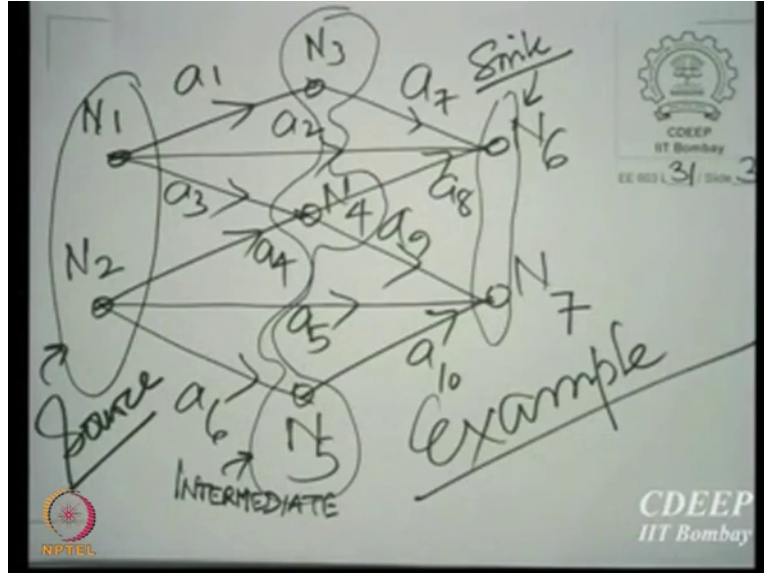


Now, the approach to realization is essentially by using what are called signal flow graphs and we need first to define a signal flow graph in a slightly more formal way than we have done before. A signal flow graph is a collection of nodes and directed edges. So, you know even among nodes we have different kinds of nodes, we have source nodes, we have intermediate nodes and we have sink nodes.

So, for example, nodes from which all the edges go outwards that means nodes which provide to other nodes only are called source nodes, nodes which have edges coming in as also going out are called intermediate nodes. So, neither source nor sink they have some edges which come in which bring material to the node and some edges which carry material away from the node. Subsequently, we have what are called sink nodes.

Sink nodes are nodes to which edges come but from which no edges leave and in fact the word source and sink are fairly suggestive, source means that which gives, sink means that which consumes, intermediate is neither source nor sink. Now, it is best to understand this by taking an example. So, let us take an example of a signal flow graph in which we have all the three kinds of nodes, source nodes, intermediate nodes and sink nodes.

(Refer Slide Time: 04:27)



So, we have this signal flow graph that we have constructed here and what we have done is to construct 1, 2, 3, 4, 5, 6, 7 nodes and we have several edges we have this edge this this one all these. For each edge we have given a multiplier on the edge you know we have the edges as we remember, an edge carries a multiplier on it.

So, good way to understand nodes and edges is that nodes are like stations at which material is deposited or from which material is taken edges are like trucks, which carry away material from one station to the other station. And in the truck there is some machinery which does processing on the material which is being carried.

Now, you must think of every truck that carries material away from a node as carrying the same material. So, it is not as if there are more trucks, the material is divided among the trucks, no there is no conservation law here. Whatever material is available at a station is all carried by all the trucks that leave that station.

However, as far as material being created at a station is concerned, all the edges which come in deposit their respective material all the trucks, if you please, which come in to a station deposit their material on the station and what material is seen at the station is the linear combination is the sum of all this material that is so deposited.

So, the rules are different for deposit and takeaway for deposit, all the material that is brought is added to form the material at that nodes for take away the material is taken away the same by all the trucks that leave as I want to repeat once again, there is no conservation law here. Anyway, so for example, here if you look at this signal flow graph  $N_1$  and  $N_2$  are examples of what are called source nodes.

So, these are source nodes here  $N_3$ ,  $N_4$  and  $N_5$  are examples of what are called intermediate nodes and  $N_6$  and  $N_7$  are examples of what are called sink nodes. Now, to take an example of deposit and take away all the edges  $a_1$ ,  $a_2$  and  $a_3$  carry away the same material from the edge  $N_1$  that is whatever material is being provided by node  $N_1$  is carried away equally well by these three edges.

This edge multiplies that material by  $a_1$  this one multiplies the material by  $a_2$  and this multiplies the material by  $a_3$  and of course, at  $N_3$  you have  $a_1$  times this material being deposited at  $N_6$  you have  $a_7$  times this material  $a_2$  times this material and  $a_8$  times this material coming together and being added and deposited.

So, for example,  $N_6$  is an example of a sink node here. So, it has no edges going outwards and  $N_3$ ,  $N_4$ ,  $N_5$  as I said have edges coming inward and outward if you looked at  $N_4$  for example, the material that you have at  $N_4$  is  $a_3$  times the material that you have it  $N_1$  plus  $a_4$  times the material that you have it  $N_2$  being added together and deposited to form the material at the station  $N_4$ .

So, much so then for signal flow graphs has a mechanism of representation of transmission of sequences signals or data. You see, you must remember, the signal flow graph can hold on it, either a single number or  $Z$  transform or any other entity on which an operation which the edge is capable of doing is possible.

So, hear in this context, we have only single numbers and we will use signal flow graphs later with single numbers or unique number data for realizing discrete Fourier transforms efficiently. But we will make use of  $Z$  transforms sitting on the nodes or at the stations to realize discrete systems, as we will do over the next few lectures.

(Refer Slide Time: 10:29)

The image shows a whiteboard with a handwritten difference equation. The equation is  $y[n] = \sum_{l=1}^N a_l y[n-l] + \sum_{m=0}^M b_m x[n-m]$ . The first term is circled in blue, and the second term is circled in red. There are some scribbles and a large 'R' at the bottom right of the equation. The whiteboard has logos for CDEEP IIT Bombay and NPTEL in the corners.

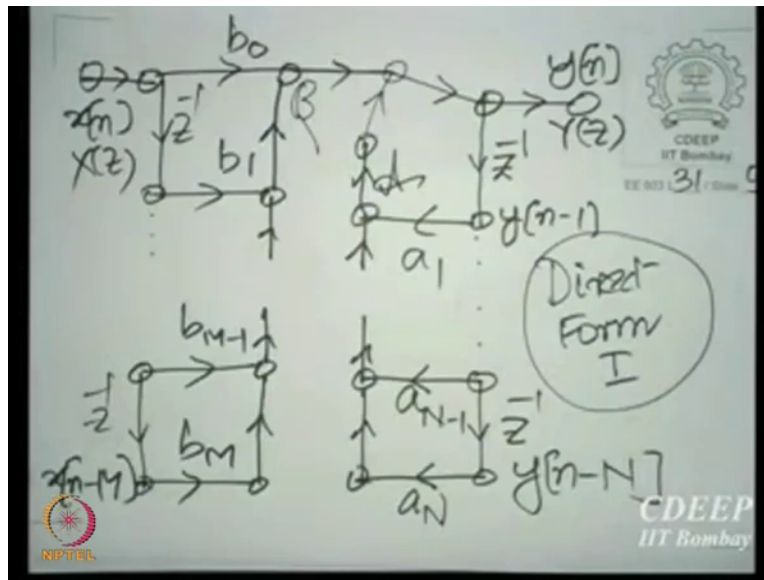
$$y[n] = \sum_{l=1}^N a_l y[n-l] + \sum_{m=0}^M b_m x[n-m]$$

So, you see, let us then come to a specific realization that we want to do, namely the rational causal not necessarily stable though, rational causal system that we written down a few minutes ago. Let us, first write down a difference equation, linear constant coefficient difference equation that realizes that rational causal system.

So, there the output  $y$  of  $n$  is related to the input, I mean the input accent  $x_n$  and it is past samples and also the past samples of the output. So,  $y$  of  $n$  is a combination of all the input samples from the most recent one to the sample capital  $M$  away a proper linear combination which we have represented by script  $B$  here.

And plus a linear combination of the past capital  $N$  outputs starting from the output one sample away. So, there are  $N$  terms here and capital  $N$  terms here, and capital  $M$  terms here. This is the difference equation that describes that rational causal system that we had in the very beginning of this lecture.

(Refer Slide Time: 11:57)



$$y[n] = \sum_{l=1}^N a_l y[n-l] + \sum_{m=0}^M b_m x[n-m]$$

Now, we want to draw a realization of that system by using adders, multipliers and delays. So, you see, what we will do is to realize the feedback part first or the feed forward part I am sorry. So, we will first realize this part, we will take the sum of the input samples and put that into a structure where it is very easy to see what that structure says.

It essentially says take  $b_0$  times the input, delay the input by 1 take  $b_1$  times that important and so on take  $M$  such delays in cascade and you get  $x[n]$  here you get  $x[n-1]$  here and  $x[n-M]$  here, take  $b_0$  times this plus  $b_1$  times this plus  $b_{m-1}$  times just the last penultimate one and finally  $b_M$  times  $x[n-M]$ .

And add them two at a time, we have agreed that we will like to make use of two input adders to be uniform in our structure. We do not want to of course, nothing stops us from using adders which have more than two inputs but it is always desirable to use the same kind of unit repeated again and again in a realization.

So, we prefer to make use of two input adders instead of using arbitrary number of inputs at each adder. So, with that, then we take two at a time and add them. So, we have  $b_m$  times this plus  $b_{m-1}$  times this being added in the first node and you can continue to add one at a time until you reach  $b_0$  times  $x[n]$  here.

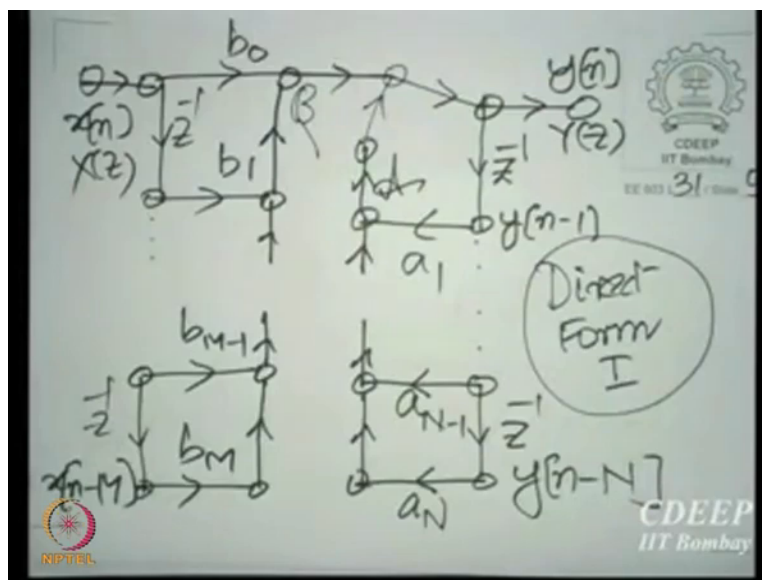
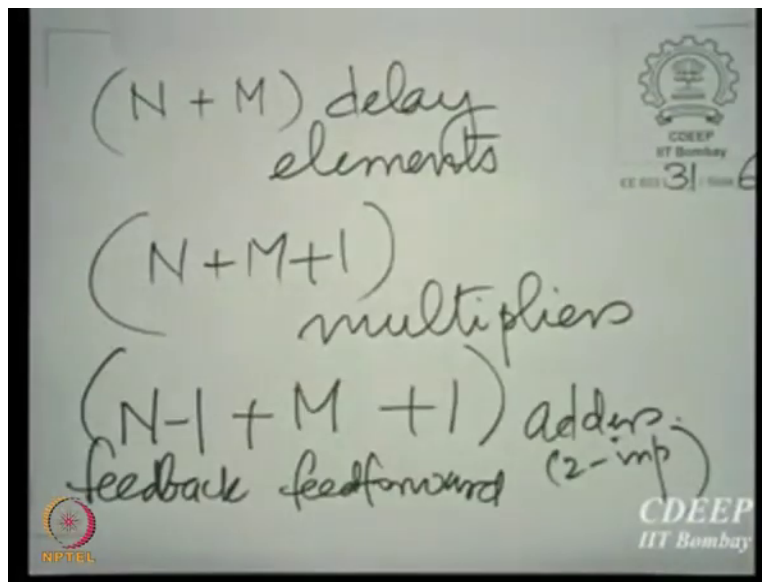
And together all this gives you this part of the sum here  $b_m x[n-m]$  sum for small  $m$  going from 0 to capital  $M$  this part is obtained at this node here. Now, for the other part of the output, let us assume that we have the output somewhere here generated we operate capital  $N$  delays on that output. So,  $n-1$ ,  $n-2$  up to  $n-N$ , we operate capital  $N$  delays on this output to give you  $y[n]$   $y[n-1]$  all the way up to  $y[n-N]$  here.

Now, as this equation suggests you need to multiply  $y[n-1]$  by  $a_1$   $y[n-2]$  by  $a_2$  and so on when you reach  $y[n-N]$  being multiplied by a capital  $N$  and therefore, you have these multipliers  $a_1$  up to  $a_{N-1}$   $a_N$  being operated on these delayed versions of the output and then we sum them two at a time.

So, we sum this and we sum this we get the first sum, keep on doing this until you take the factor  $a_1$  times  $y[n-1]$  and finally what you generate here is the expression script  $A$  that you have here. So, we have script  $B$  generated this script  $A$  being generated here. And if you add these two, it generates  $y[n]$  for you.

So, we are in good shape, we have generated  $y[n]$  back again by using what is called a feed forward section and a feedback section, this is the feedback section and this is the feed forward section here.

(Refer Slide Time: 16:15)



Now, how many elements of different kinds have used in this realization, it is very clear that we used ' $N+M$ ' delay elements ' $N+M+1$ ' multipliers and ' $N-1$ ' adders in the feedback path and  $M$  adders in the feed forward path and then 1 adder overall to add the feedback and the feed forward combinations resulting in ' $N-1+M+1$ ' adders. And these are assumed to be two input adders, remember.

Let us, just quickly convince ourselves on this you have  $M$  capital  $M$  delays here capital  $N$  delays there. So, ' $N+M$ ' delays you have capital  $M$ , adders to input adders here, because you have all the way from  $b_0$  to  $b_M$ , you have capital ' $N-1$ ' two input adders here and you have one



more here. And of course, as far as multipliers go you have M+1 there and capital N here. So, that is an explanation for the numbers that we put down here.

(Refer Slide Time: 18:07)

$$H(z) = H_A(z)H_B(z)$$

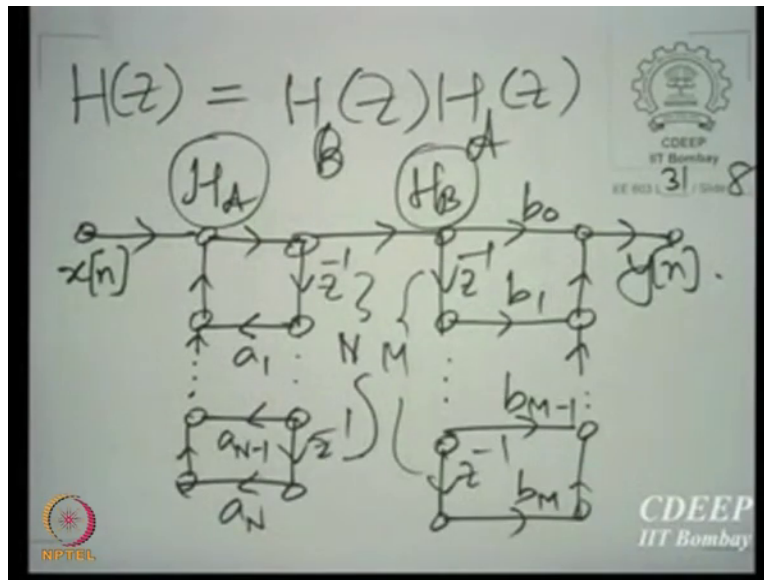
$$= \left\{ \frac{1}{1 - \sum_{l=1}^N a_l z^{-l}} \right\} \left\{ \sum_{m=0}^M b_m z^{-m} \right\}$$

interchangeable

Now, we make an observation which will help us actually reduce the amount of hardware or software that we are using in realizing the system. In fact, we noticed that what we have done is to decompose the system function into two parts a so called feedback part and so called feed forward part and we operated the feed forward part first and then the feedback part because we have written  $H[z]$  in this following cascade decomposition format.

The numerator has been put first  $\{1/1 - \sum_{l=1}^N a_l z^{-l}\} \{ \sum_{m=0}^M b_m z^{-m} \}$  and the denominator has been put next and we may treat  $H(z)$  as a product of the numerator system function and the denominator system function. Obviously, this is a product of two functions of  $Z$  and that product is interchangeable multiplication in the  $Z$  domain is also of course commutative. So, we could put this feedback structure first and the feed forward structure next.

(Refer Slide Time: 19:11)



Handwritten equation illustrating the transfer function  $H(z) = H_A(z)H_B(z)$  in direct form 1. The numerator is a sum of terms  $b_m z^{-m}$  for  $m=0$  to  $M$ . The denominator is  $1 - \sum_{l=1}^N a_l z^{-l}$ . The equation is labeled "interchangeable". The diagram is watermarked with NPTEL and CDEEP IIT Bombay.

And let us indeed do that in the architecture here. So, if you had only the feedback structure, then you would have a structure like this realizing it in direct form 1, you would first put capital N delays, you would multiply the first output of the delay the output of the first delay by  $a_1$  and so on up to a capital N.

You add two at a time and then finally complete the sum with the input to produce the output here up to here. So, this realizes the numerator so and sorry the denominator, this realizes the denominator this realizes,  $H_A(z)$  if you recall,  $H_A(z)$  is this so here you have a numerator of 1

and a denominator given by this. And if you use the structure that we just described, this is the realization with that structure.

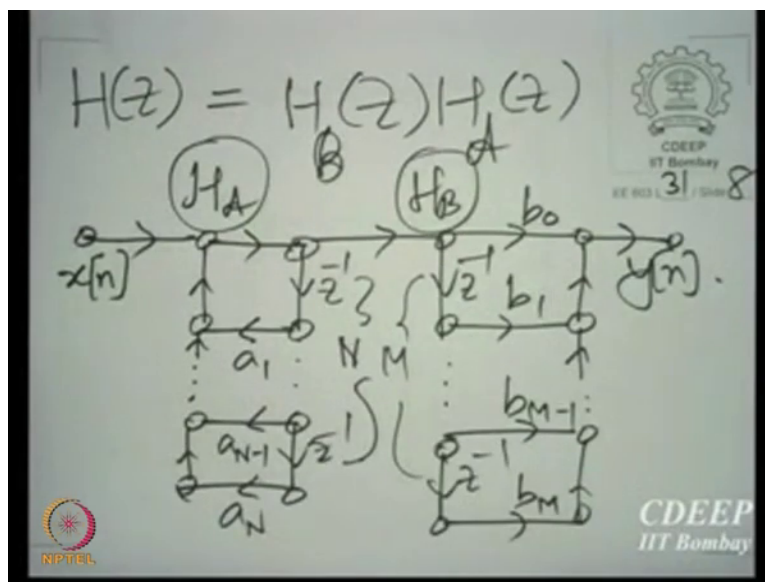
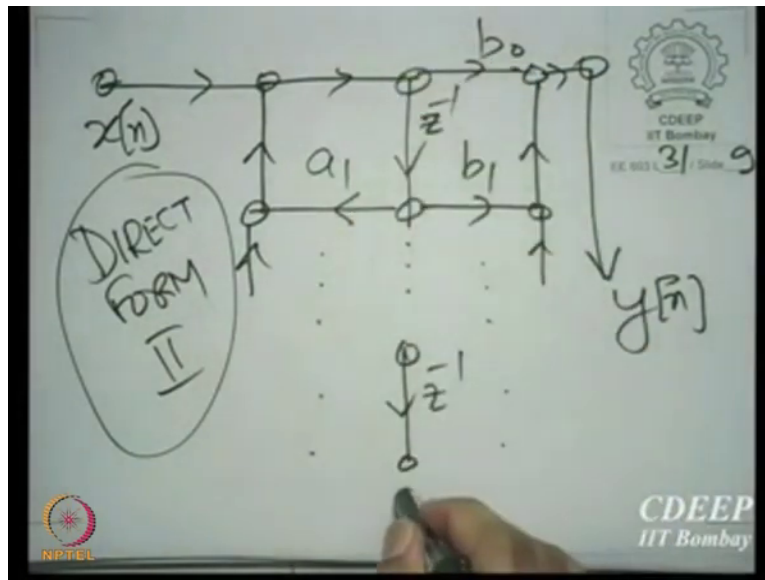
Together with that you have an  $H_B(z)$  also being realized, an  $H_B(z)$  is realized in this way, you would simply put a cascade of capital  $M$  delays, multiply subsequently by  $b_0, b_1$  up to  $b_M$  and then add 2 at a time to produce the output. And this is indeed the final output here. Now, when we put the system function in this form, something becomes evident immediately.

What you have here? Now let us use the rules of signal flow graphs. So, you have a station here, what is at this station is essentially what is this station carried with a multiplier of one. So what is located this station is the same as what is located at the station, there is no difference. These are only arrows going outwards, if you notice.

And therefore, what we have here is this delayed by one sample, what we have here is this delayed by one sample and if you look at it, these in sequence must carry the same material to this material must be the same, this material must be the same. Similarly, if you go step by step, one, one delay downwards, the corresponding pair of nodes, the corresponding pair of stations must carry the same material. So, we are in fact wasting stations here and delays as well.

You might as well have picked because each station has the capability to provide as many outgoing edges as you desire, so there is really no need to keep two stations carrying the same material all the time. And therefore, all that we need to do is to fuse this train of delays and this train of delays. And you would do that by looking at which is larger capital  $N$  or capital  $M$ , putting a stream of delays with the maximum of capital  $N$  and capital  $M$ , delays on that stream and tapping off the outputs of the delays one after the other.

(Refer Slide Time: 22:52)



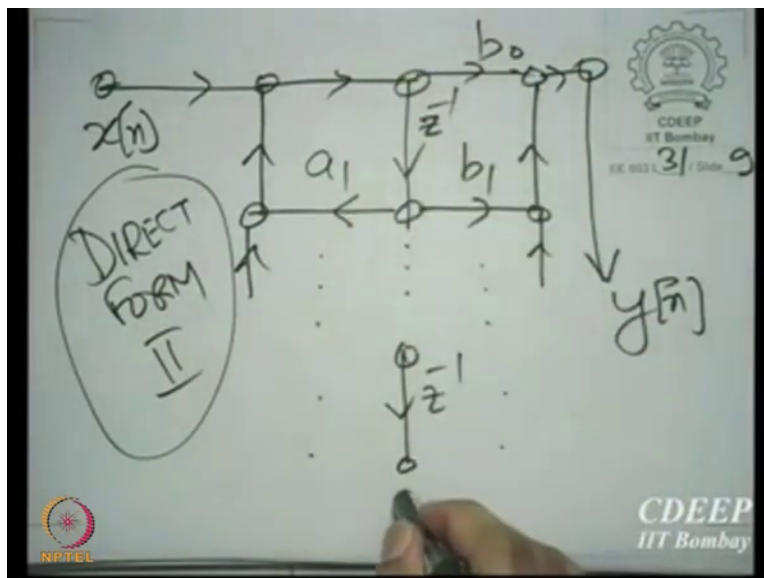
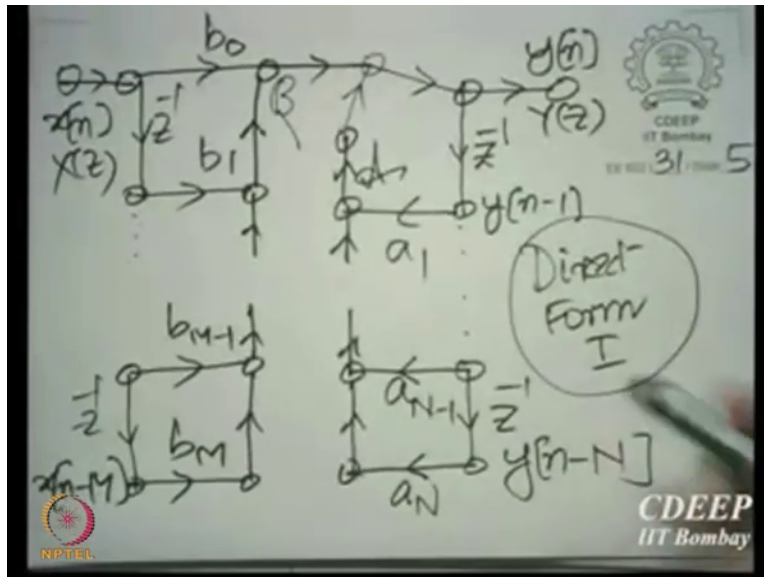
And that leads to a structure like this. Note carefully what is happening here, you have the middle node, as you did here, you have this middle node, the middle node, these are really the same is placed here, you delete these one sample at a time. And you have a stream of delays with maximum of  $n$  and  $m$  delays in the stream, you might taken  $a_1$  times what comes out to the first delay and fed it backwards and you have taken  $b_1$  times what comes out to the first delay and fed it forward.

And you keeping on doing this  $a_1$   $b_1$  here  $a_2$   $b_2$  next and so on. And except at the top branch you would need to multiply this by  $b_0$ . And the rest of it is of course the same does not change. Now

you see obviously, here we have kept the number of multipliers the same, there is no change the number of multipliers.

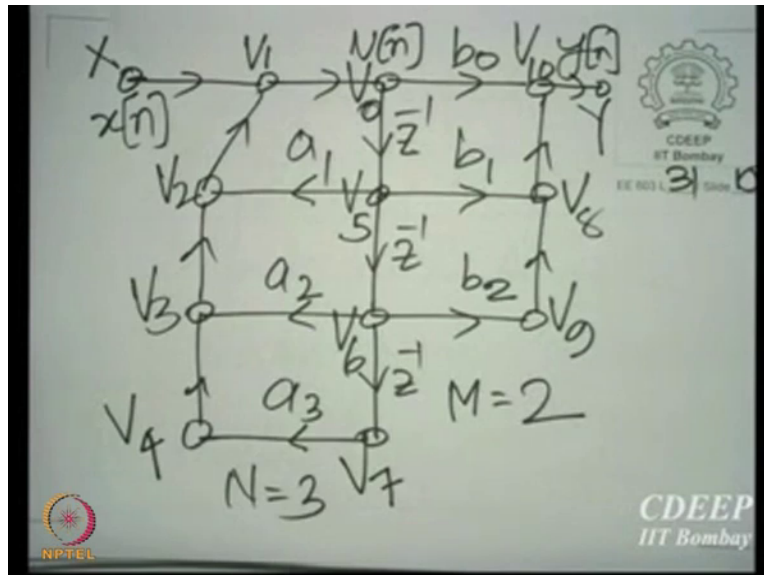
In fact, there is no real change in the number of adders as well. What has changed is the number of delays. But yes, quite a bit from 'N+M' delays the maximum of N and M delays and therefore, this architecture is definitely more economical in terms of hardware or software requirement. This is called direct form two realization this realization is called direct form two.

(Refer Slide Time: 24:29)



As against the earlier realization, here which we call direct form one so just to flesh both before you this is direct form one where you have all these additional delays and this is direct form two where you have economize on the number of delays.

(Refer Slide Time: 24:51)



Now let us, take an example. In fact, let us put down an example where you have 'M = 2' and N=3 and draw the direct form two architecture. So, you can see very clearly that you would have a node here you would have so, you see if 'M = 2' and 'N = 3' than their maximum is 3 and therefore you need 3 delays in cascade in a stream you tap off  $a_1$  times this  $a_2$  times this and  $a_3$  times this and add them two at a time and finally, add the input to produce the intermediate node.

Subsequently, we multiply this by  $b_0$  this by  $b_1$  and this by  $b_2$  and add them again two at a time multiply this by  $b_0$  and of course, so, here you have  $y[n]$  being generated by the feed forward path. So, feed forward path is here, the feedback path is here. And let us give names to each of these nodes in turn,  $V_1, V_2, V_3, V_4, V_0$  here we begin from this node  $V_5, V_6, V_7, V_8, V_9$  and finally  $V_{10}$  and then we have the input node X and the output node Y.