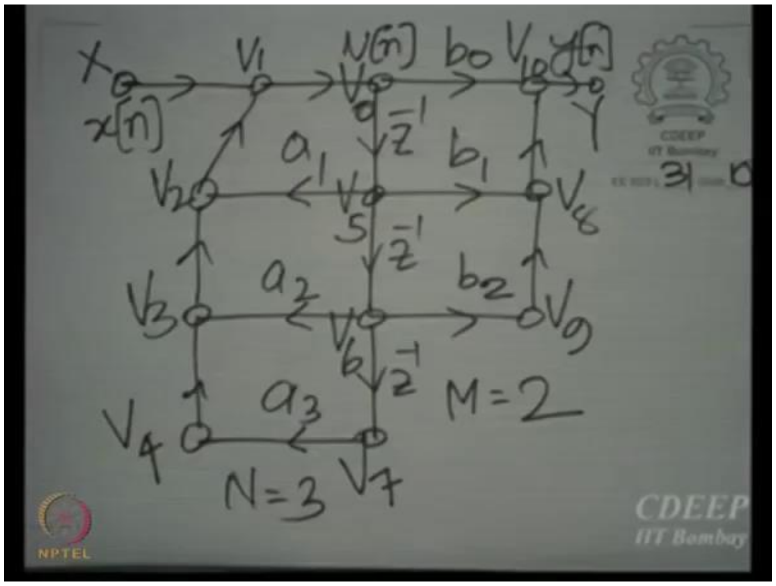
Digital Signal Processing & Its Applications
Professor Vikram M. Gadre
Department of Electrical Engineering
Indian Institute of Technology Bombay
Lecture 31 b
Obtaining Pseudo Assembly Language Code

(Refer Slide Time: 00:15)



Now, you see having drawn a signal flow graph, the next task is to realize that signal flow graph in the form of an algorithm or in the form of what is called pseudo assembly language or pseudo code. We need to be able to program structures, we cannot be satisfied simply with drawing them. So, to program the structure, we need to evolve what is called a generic loop.

You see, remember, you are going to have input samples coming in all the time and you are

Going to have output samples going away all the time cycle by cycle for every sampling instant

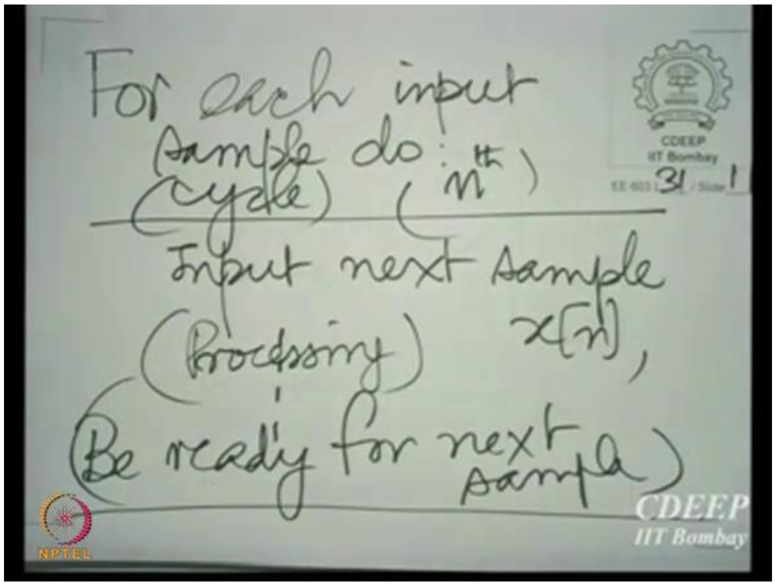
You have an input sample coming, you have some processing being done with that input sample

And a few pass samples of the output a few past samples of input and you have an output sample

Being generated. And all this work for that output sample has to be done within the time

Available to you between two samples of the input arriving.

(Refer Slide Time: 01:36)



For each input sample do: (cycle) (n^{th})

Input next sample $x[n]$,

(Processing)

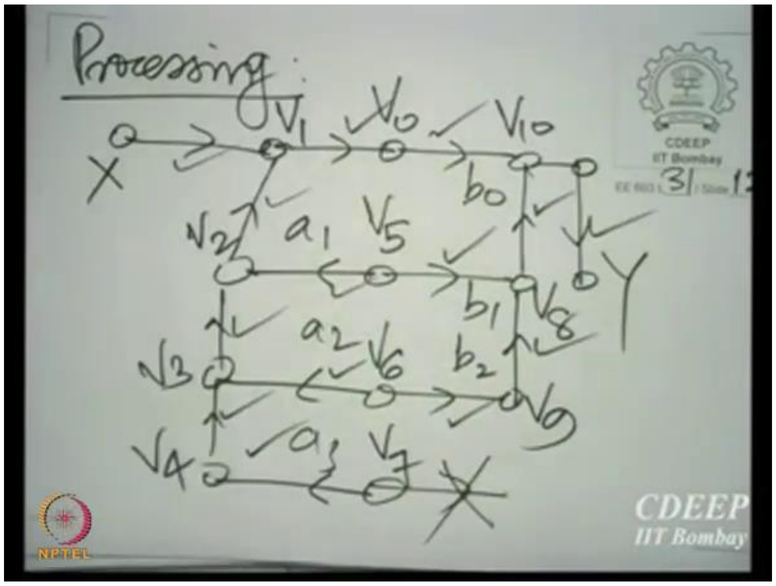
(Be ready for next sample)

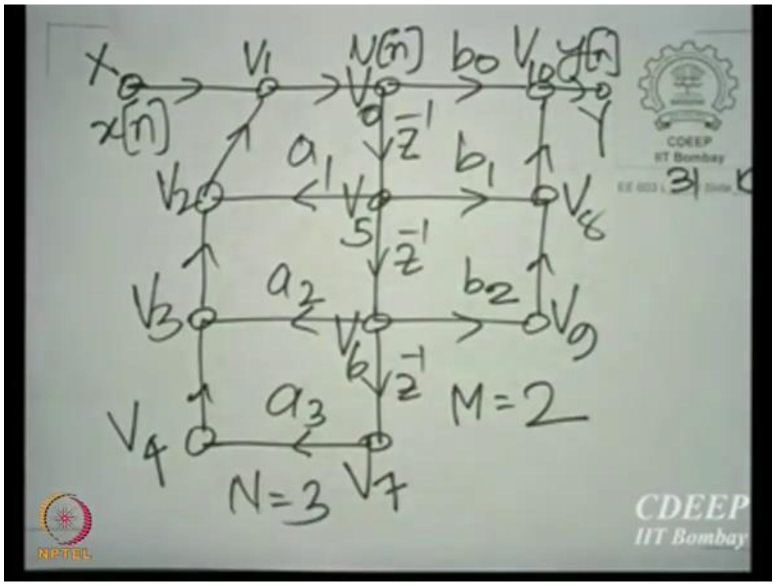
CDEEP
IIT Bombay
EE 603 L 31 / Slide 1
NPTEL
CDEEP
IIT Bombay

So, this is the generic loop that you would require to realize that signal flow graph for each input sample, do all the following that is the in the nth sampling cycle, input the next sample, let us call it x[n] carry out some kind of processing and be ready for the next input sample to come, that means at the end, so there is a loop operating for every input sample here.

After you input the next sample, you have to make sure that you are ready with whatever was required for that input sample to be process before that sample came in that is why we are saying be ready for the next sample.

(Refer Slide Time: 02:25)





So, let us again take the same signal flow graph, and let us evolve a process to write down a pseudo code for that signal flow graph. So, let us flashback that signal flow graph for you. Now, the rule is very simple, you see, what really are delays. Delays, as you see here are mechanisms for preparing yourself for the next sample.

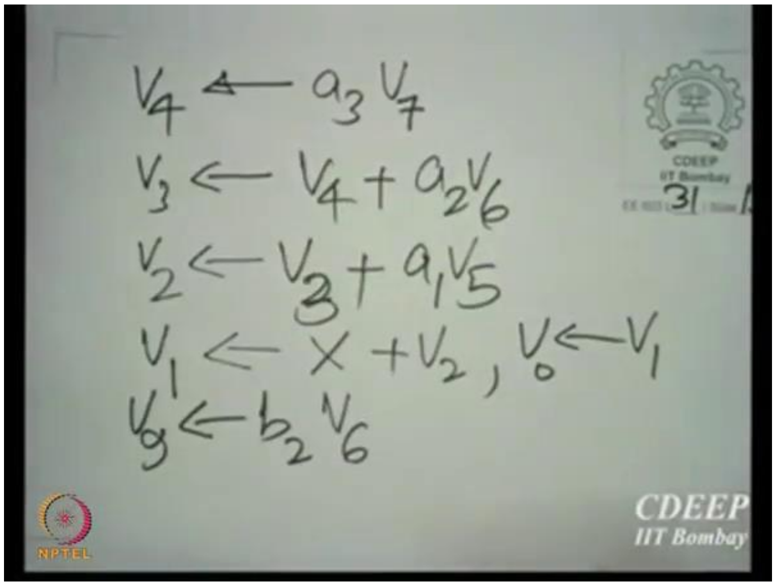
So, the idea is very simple. How do you decompose the signal flow graph into an algorithm or into pseudocode? First, think of the delays as having been removed. So, take a pair of scissors notionally and cut off these threads bearing the delays cut off the edges bearing the delays. And that would take you to a signal flow graph that looks like this as you see, compare this so we have essentially retained V_0, V_5, V_6, V_7 all the nodes, we retain all the nodes, but we have cut off all the edges carrying delays.

And the multipliers are retained as they are. Now, we need to talk about being able to realize a signal flow graph. You see, once you have done this and you have generated a reduced signal flow graph without the delays, the signal flow graph is realizable only if it has no loops that are lefts. Loops are a succession of edges, which begin from one node and which end at the same node.

So, if you looked at the original signal flow graph here and if you looked at this succession of edges, this one, this one, this one and this one, they form a loop, you come from here you traverse these edges and reach back then there is another loop here you traverse this edge then this one, then this one, then this one, this and this that forms another loop. In fact, there is a third loop which involves a_3.

However, when we cut these delays off, we have no loops left at all you can verify that there is no loop in this resultant signal flow graph. And that gives us the secret of being able to realize a signal flow graph, no loops which are devoid of delays. If you have a loop, which contains a delay which does not contain a delay, there is a problem. If you have a loop that contains a delay it will get cut when you cut the delay.

(Refer Slide Time: 05:45)



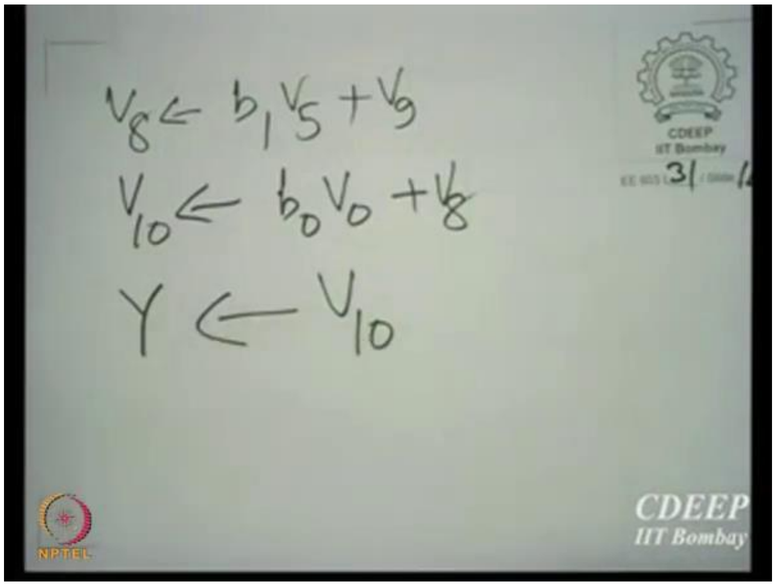
$V_4 \leftarrow a_3 V_7$

$V_3 \leftarrow V_4 + a_2 V_6$

$V_2 \leftarrow V_3 + a_1 V_5$

$V_1 \leftarrow X + V_2, \; V_0 \leftarrow V_1$

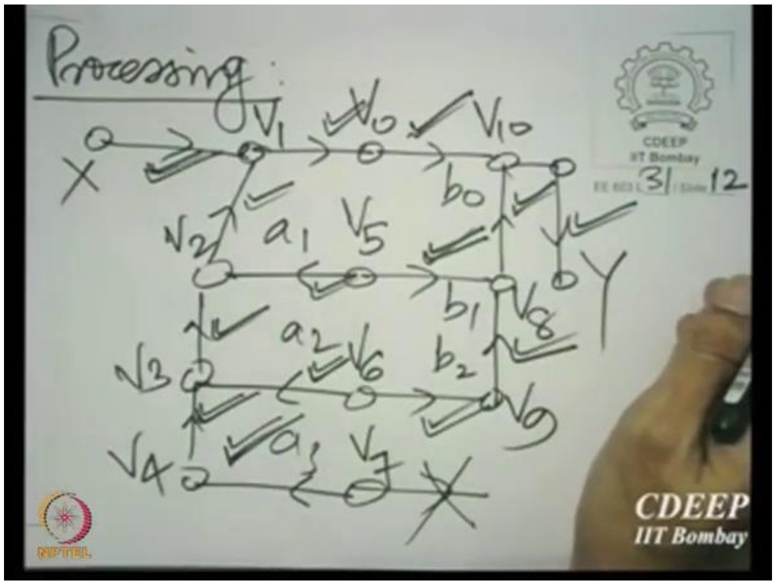
$V_9 \leftarrow b_2 V_6$



$V_8 \leftarrow b_1 V_5 + V_9$

$V_{10} \leftarrow b_0 V_0 + V_8$

$Y \leftarrow V_{10}$



Now, let us translate this reduce signal flow graph into a sequence of steps. Now, what we need to do is to identify what are called the 1st level nodes that is nodes which can be calculated first, nodes which can be calculated next and so on, step by step. So, once we have a signal flow graph like this, we have to identify the effective source nodes in the signal flow graphs you begin from the source and go to the sink that is a universal rule.

If you have no loops in a signal flow graph, the rule is very simple. Begin from the sources, there could be multiple sources and go to the sinks step by step. So, as you move from the sources, you move one step, you get the first level nodes, you move one more step, you get the second level nodes. And you can keep doing this until you reach the sink.

Now, different sink can be of different levels. Some sink might be reached in two levels. Some sink may require five levels to be traverse before they reached. Whatever it be, the level is clear from the sources once you start propagating the material then you encounter loads with increasing levels step after step.

So, for example, if you look at this reduce signal flow graph, all of these V_5, V_6 and V_7 are examples of source nodes, they have nothing coming into them and so is X. So, what we do is to take the material on V_5, let us write down. So, you see we could begin in fact, it is easiest to begin with V_4.

So, let us take V_4 and put on it a_3 times V_7 first, so that takes so let us start ticking things. So, we have taken care, if you have to take care of edges one after the other. So, ($V_4 = a_3 * V_7$) . The next step is to compute V_3, V_3 is essentially whatever is the material on (V_4+a_2*V_6). So if taken care of these two now. So, let us put a double tick on them.

The next step is to generate V_2. So, (V_2=V_3+a_1*V_5) following which we can generate V_1 (V_1 =V_2+X) essentially. So, (V_2+X =V_1) and V_0 as you can see simply is the same as V_1 V_0 takes what V_1 has on it, we have taken care of these edges now. Now, we can take care of the edges, which go forward.

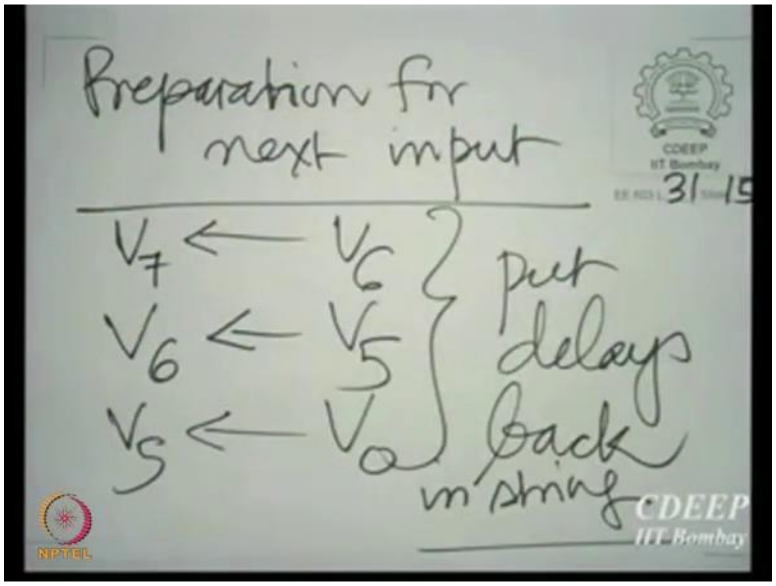
So, we can again begin with V_9. So, we because you know we have additions to be performed on the way so we should begin with the lowest stage here. So, (V_9= B_2*V_6) and then we have V_8 which is whatever is on (V_9+$b_1$$V_5$),b1 times what is there on V_5 plus V_9.

Subsequently we have V_{10} being generated (V_{10}=V_0*b_0 +V_8). So, V_8 plus b_0 times whatever is on V_0. And finally, (Y=V_{10}) and accepts it as it is. So, if taken care of this edge as well. Now, you must check that you have taken care of every one of the edges and you have done that here.

And this is where we end in terms of the program or the pseudo code. This is the last step. So, here, once we have done this, we have completed the computation. But we need to prepare for the next sample that is not all. You see, now you have cut the delays. What do you mean by cutting the delays? You did away with the effect of the delays. What is the effect of the delay? The effect of the delays is to keep with you what the previous output was before the delay.

And when you have a string of delays, you need to operate the last in the string first. Because the oldest sample needs to go away and one newer, one step newer sample needs to come to the oldest location so you need to operate a string of delays with the last one that is the most the greatest delay being operated first.

(Refer Slide Time: 11:41)



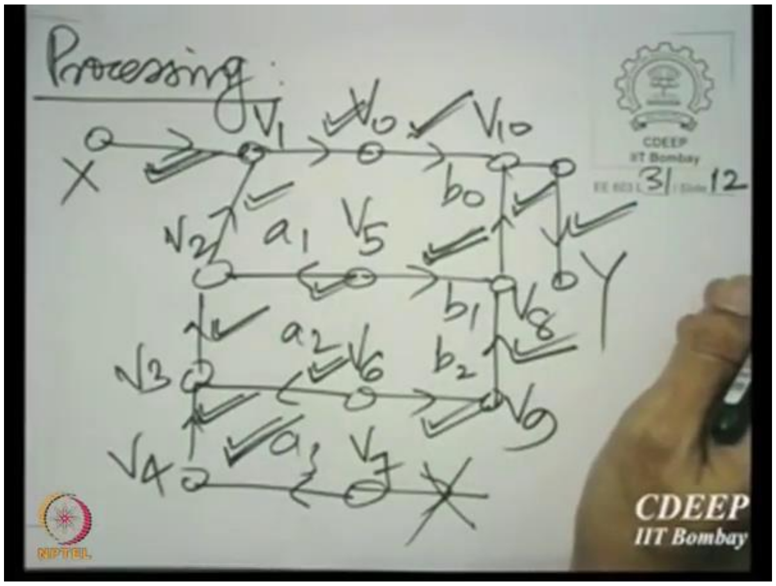
Preparation for next input

$V_7 \leftarrow V_6$

$V_6 \leftarrow V_5$

$V_5 \leftarrow V_0$

put delays back in string



So, that leads us to the step of preparation for the next input. How do we prepare for the next input look at the string of delays, the string of delays would take you from V_0 to V_5 ,V_5 to V_6 ,V_6 to V_7. And in this string, this one is the last delay and therefore you need to put at V_7 what you had at V_6. This is operated first, you need to put at V_6 what you have at V_5 and you need to put at V_5 what you have at V_0.

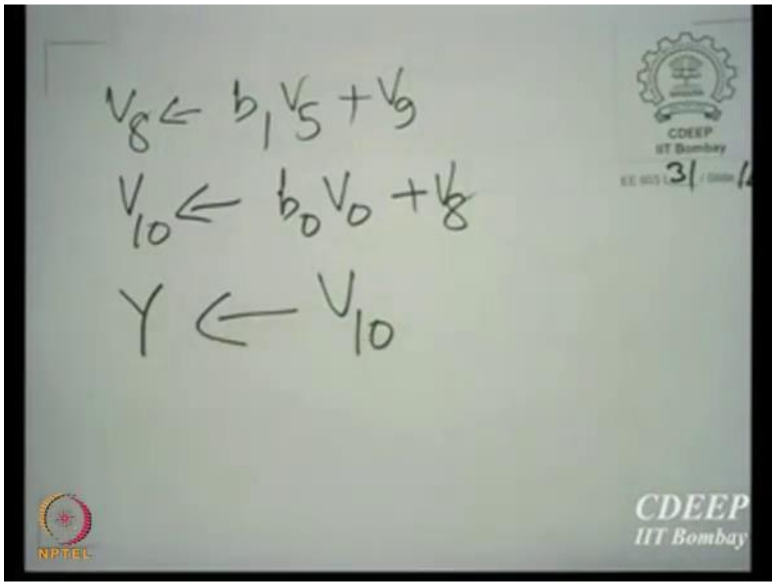
So, this is the preparation that you need to do for the next sum. In a sense, preparation for the next input means putting the delays back in the string where they were. That is what we are saying here, put the delays back in the string. Now, having put the delays back in the string, we have actually completed our job. Because we have written the pseudocode we now have a pseudo assembly level language program to realize the signal flow graph and we can convert that into whatever language we desire.

There are different languages that can be used for digital signal processing. There are different processors each have their own variants of instructions, but once we understand this basic signal flow graph and the manner in which is converted into an assembly level program, we can without too much of difficulty move from one language to the other.

You see it is very clear that just as the signal flow graph implies a hardware realization. It also implies a software realization and both of them with resources in the hardware realization resources being specifically elements like delay elements two input adders, multipliers and so on. In software, resources means computations or operations.

So, each time you make an assignment, you are doing an operation delays are like assignments, one thing is assigned another and delays operate in the end of each loop. Two input adders are indeed operations both of assignment and of summation. So, in fact, the way we wrote the program here, we did a multiplication and then we combined a multiply with an add. So, you know you have multipliers and adders being taken together in statements, you know if you remember let me back example statements for you.

(Refer Slide Time: 14:41)



$$V_8 \leftarrow b_1 V_5 + V_9$$

$$V_{10} \leftarrow b_0 V_0 + V_8$$

$$Y \leftarrow V_{10}$$

So, these are examples of statements we had, we had a multiply and then add that kind of thing. So, multiply plus accumulate is a typical kind of operation that you need in discrete time signal processing, multiply, accumulate, MAC, multiply and accumulate. Now, two input adders or edges that carry multiplications on them require multiply add kind of operations, delays essentially require assignments of course multiply and accumulate also involves an assign, you cannot do without an assignment there.

So, with this, then we have got a reasonable insight into the kind of processes that we need to convert a signal flow graph into a realization either in hardware or software. Now, in the subsequent lecture, we would actually look at different forms of realization, we looked at direct form one and direct form two.

And we have also convinced ourselves how we can write down a signal flow graph for direct form one and direct form two, but now we are going to have other alternate structures, which can also realize the same system function specifically, we would look at cascade structures, parallel structures, cascade and parallel structures and then the lattice structure. We shall do this beginning from the next lecture onwards. Thank you.