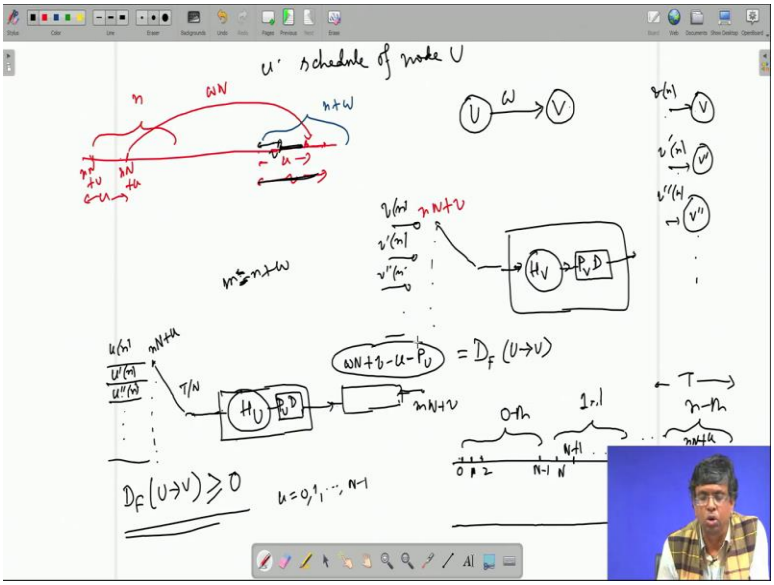
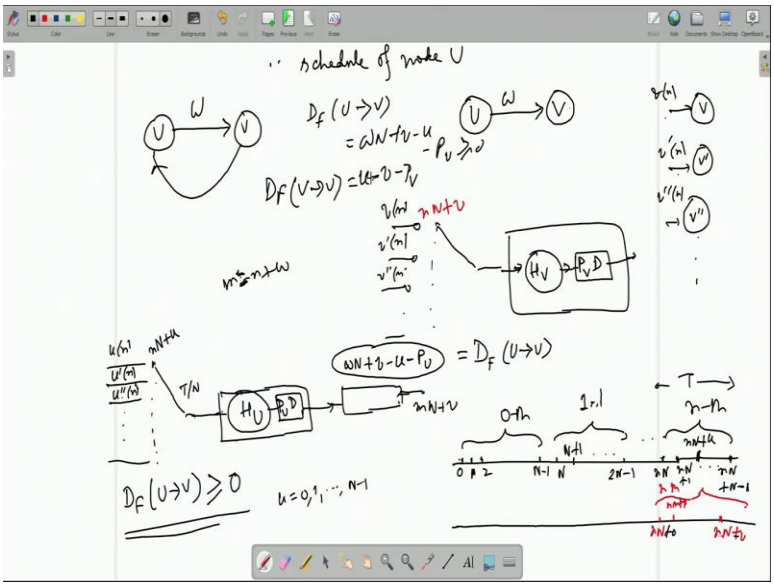
VLSI Signal Processing
Professor Mrityunjoy Chakraborty
Department of Electronics and Electrical Communication Engineering
Indian Institute of Technology Kharagpur
Folding Examples: IIR Filter
Lecture 22

(Refer Slide Time: 0:29)



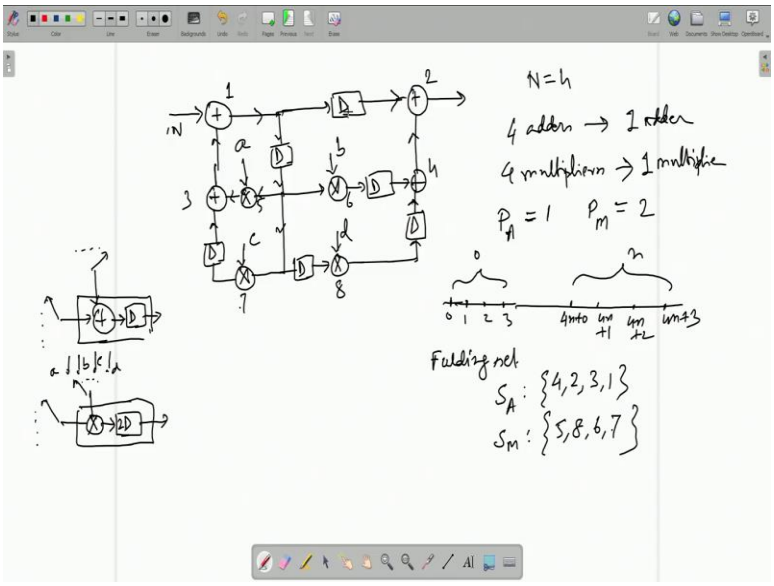
Now why this condition is important, obviously, you know you cannot have delay quite negative but there are minus signs. You can say that why not do this way that I will choose small v to be large, small u to be small, so this is positive as if WN is positive, PU may not be high, so for always positive this is not always durable.

(Refer Slide Time: 0:55)



Is because if we have a loop, if we have a loop so much w delay and reverse-path no delay, for u to v you have got, which is WN plus small v. Suppose this is greater than 0 because small v is large, small u is small, but the reverse-path w there is no delay so w is 0. So Df V to u there is no delay in the reverse path so this is 0, then you have got u because the destination is U, destination node is U, capital U and therefore in schedule volume come first with the positive sign then v then minus PV. So the small v was large and u is small here it is the opposite. So you can try to get this positive by choosing large v or small u, but that will make this negative. So you know there may be more complicated loops and all that, that is why it is not very easily done but this is an additional constant that comes. Now let us take an example of an IIR filter and let us try to fold it. Let me see how far you can go today, if we cannot complete we can take it to the next class.

(Refer Slide Time: 2:29)



This is adder 1, a means the multiplier with a built-in constant a, b means a multiplier with a built-in constant b, same for c. So adder 1, adder 2, adder 3, adder 4, Multiplier, I am giving them 5, 6, 7, 8, folding factor N is 4 that is 4 adder to be replaced by 1 adder, to be replaced by 1 adder, 4 multipliers to be replaced by 1 multiplier. Further it is given that every adder has got PA that is the number of pipelining latches for forward path retiming of the adder hardware unit like PU and PV earlier it is PA, A for adder. So PA is 1, there is only one delay.

So, there is an adder in order to make it work at a faster clock I am just doing a forward path retiming, pipelining just by one latch. So output will be delayed version of the input of the original output and delay is one cycle. In the case of multiplier, multiplier requires more complex you know hardware and all that, takes more time. Therefore, I need more number of latches to pipeline it. So Pm number of multiplier pipelining latches to pipeline one multiplayer unit that is given to be 2, because it takes more time. So I need to, I need more delays to cut it which is given.

Also given r, the schedules of the 4 adder and 4 multipliers that is originally if this is a timing diagram, nth clock who start at 0 they are dividing into 4, so 0, 1, 2, 3 dot dot dot dot 4, 5, 6, 7 like that. So nth will be like small n into capital N, capital N is 4 so small n, so 4n plus 0,4n plus 1,4n plus 2, 4n plus 3, they are the four system cycles. One will be, as I am folding all the 4 adders into 1 adder. So that 1 adder hardware unit with pipeline latch, built-in pipeline latches

equal to 1 that will take up the job of one of the adders in one clock, another adder in another clock, another added in another clock and so on and so forth. Those will be which clock that will give the schedule of that particular adder.

So, this is defined by a notation S is called folding set. It will be very clear folding set say S for set, A for adder, I have got 4 adders 1, 2, 3, 4. So within curly bracket, it is given like this. I am taking the example from (()) (08:28) book, so 4, 2, 3, 1. Adder 4, adder 2, adder 3, adder 1, which means at first, in the first system cycle of any nth input cycle that adder unit, hardware unit adder which I am going to employ that will take up the job of 4, adder number 4. The next one at 4n plus 1 that adder unit will, that unit hardware unit will take up the job of adder number 2, at 4n plus 2 it will take up the job of the adder number 3 and at 4n plus 3 it will take up the job of adder number 1.

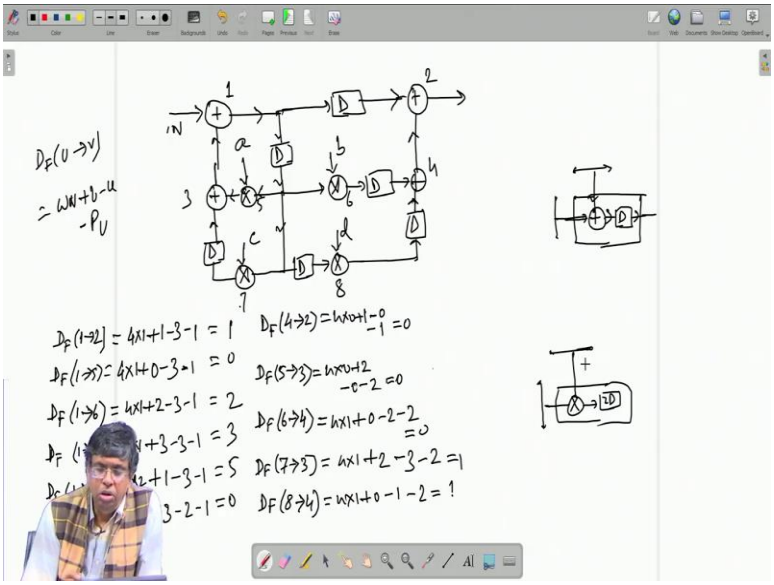
So just this was the order at 4n plus 0 that adder unit will take up the job of adder 4. What is the adder unit, like here we will have adder, I am showing the pipelining latch D separately here, adder has got two inputs, is a two input one output device. This will be switched there are multiple input lines, multiple input lines. So it will touch them at various system clocks. 4 system clocks are possible. So only four schedule, in one schedule it will touch one particular line here and one particular line here and in another schedule it will touch another line here, another line here, like that. And that way it will work for either 1 or 2 or 3 or 4. Similarly, multiplier, multipliers have got built-in constants. So multiplier units will be like this. The actual multiplier and pipeline by two latches those two latches we show separately here.

The single input and built-in constant can be a, b, c, d so again this is another input which is also switched. So there will line like a or built-in constant value c, d. So in one cycle it may touch c another cycle it may touch a and like that. So folding set Sa 4, 2 ,3, means this hardware unit adder it will take up the job of adder number 4, node 4, node 4, node 4 at 4n plus 0 the first system cycle under nth of under every nth Watt cycle. Then at 4n plus 1, this unit will take up the job of node 2 adder 2 and so on and so forth. Similarly, for the multipliers, this particular example from (()) (11:24) book says folding set to be followed is 5, 8, 6, 7.

So what you have to do is, we have to find out the delay values, the Df. We have to finally fold but before that, we need to find out, like we will imply one hardware unit for all the 4 adders, one multiplier unit for all the 4multipliers. So what will be the delays for various node to node

edges here like this node has got an edge like this, adder to multiplier with one delay or adder to adder with one delay, likewise. So we will use that formula of the folded delay formula and just mechanically calculate them.

(Refer Slide Time: 12:24)



Alright, start at one, Df node 1, node u to v, 1 has got 1 is to 2, but between 1 to 2 how many delay, one delay. So formulae we remember Df, I am rewriting for your sake, was, v is the schedule of the destination node, u is the schedule of the source node, Pu is a number of pipelining latches of the source node. W was the original input cycle delays between u to v, capital N is the folding factor. So here u is 1 and v also, so here adder again adder same unit to same unit, adder to adder, but nevertheless from this that which adder will get a feedback to its input means adder to adder I have got only 1 adder unit in a final circuit.

So 1 to 2, we have got one delay, so W is 1, n is 4, so 4 into 1 plus schedule v, schedule v means schedule of 2, what is the schedule of 2 that was 4n plus 0, this is 4n plus 1. So schedule is 1, minus schedule of 1, 1 here is 4n plus 0, 4n plus 1, 4n plus 2, 4n plus 3, 4n plus 3. So schedule of 1 is 3, small u is 3, minus Pa, u is a. How many is Pa 1, so this is 1. Then see next edge 1 to 5 adder to multiplier, one delay. So again 4 into 1, WN, W is 1 here, N is 4, 4 into 1 plus schedule of 5, 5 is multiplier it is given 4n plus 5 is coming here as a first guide that is that 4n plus 0. I have got the schedule of 5 so small v is 0 minus these two do not change because it is still node 1

so schedule of node 1 is 3, we have already seen 4n plus 3, number of pipelining latches for adder is 1, so this is 0.

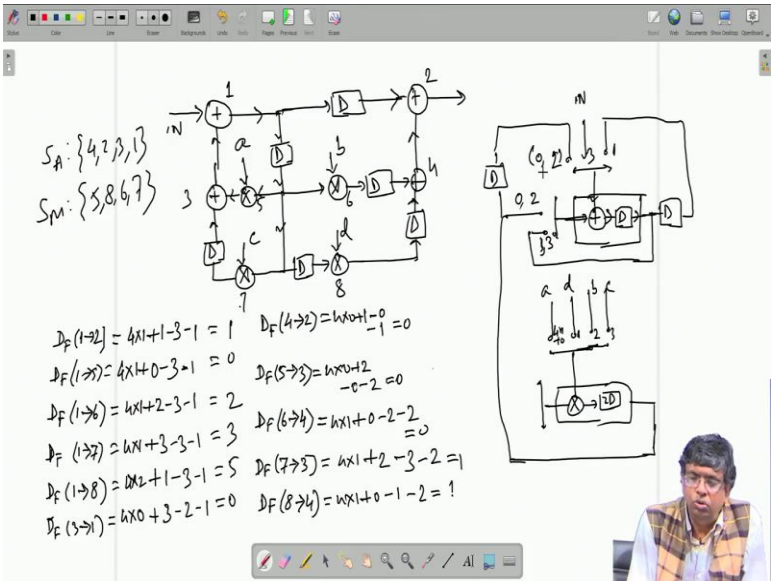
The next stage is 1 to 6, again one delay plus 6 its schedule is 2 because 4n plus 0, 4n plus 1, 4n plus 2 minus this part does not change because starting node is node 1. So node 1 schedule is 3 and pipelining latches is 1 for the adder. So this is 2, that is very routine thing. Then so I have covered 1 to 2, 1 to 5, 1 to 6. Now again 1 to 7, this has also one delay. So 4 into 1 plus schedule of 7, schedule of 7 is 3 because 4n plus 0, 4n plus1, 4n plus 2, 4n plus 3 minus this part does not change because node is 1 for all of them. So it is 3 and lastly 1 to 8, 1 to 8 has got 2 delays.

So it is 4 into 2 plus schedule of 8, schedule of 8 is 1, 4n plus 0, 4n plus 1, 1, minus of course they do not change because it is all for node 1. So 5, so all are positive here. Then start at node 2, node 2 does not have any edge to anybody so forget that. Node 3, 3 to 1 adder to adder, no delay, so 4 into 0 plus schedule of 1, which is 3 minus schedule of 3 which is 2 minus 1 because both are adder so pipelining latch is Pa is 1, so this is 0 still not negative. Then comes Df, 3 to 1 I have done, for 2 2, we have done with 3 then 4, 4 to 2. Again there are two adder, no delay. So 4 into 0 plus schedule of 2, there is 1 minus schedule of 4 that is 0, 4n plus 0 minus pipeline latch is 1 so 0. Then 4 to 2, from 4 no other edge goes so I am done with, done up to 4. So now 5, 5 is a multiplier, from 5 also only 1 edge 5 to 3 and no delay here.

So again 4 into 0 plus schedule of 3 is 2 minus schedule of 5, schedule of 5 here is 0. Now minus pipelining latches here will be 2 because source is a multiplier, multiplier has got Pm equal to 2. Sorry which there at 0, then 6 to 4, 6 to 4 there is one delay so 4 into 1 then schedule of 4, 4 is an adder, schedule is 0 minus schedule of 6 which is 4n plus 2, 4n plus 0, 4n plus 1, 4n plus 2, so 2 minus pipelining latches for the source nodes. There is multiplier here, which is 2, so again 0, so nobody is negative. Then from 6 to 4 done, next is 7 to 3, 4 into 1 because there is a delay schedule of 3, which is 2, 4n plus 2, minus schedule of 7 which is 3, 4n plus 3 minus number of pipelining latches for 7 which is Pm, that is 2 so it is equal to 1.

And lastly 8 to 4, 8 to 4 one delay so 4 into 1, schedule of 4 is 0 minus schedule of 8 which is 1 minus number of pipelining latches 2, so 1. Let us have the hardware unit for adder as I told you 2 inputs. The switched and one delaye inside. Then there is a multiplier, 2 delays inside. This input is switched, and this also switched but this is switched over a, b, c, d, so (()) (21:26) but the schedule.

(Refer Slide Time: 21:41)



Let me rewrite this schedule because we need them. SA was (()) (21:44) by this step 4, 2,3, 1 or multiplayer it was I think 5, 8, 6, 7. So, see at 4n plus 0 this is taking up the job of multiplayer 5 for the input a, a. So this is touching it at 4n plus 0. Every time I will not write 4n, I will drop it, I will just write 0. Then next is 8, 8 it is schedule is 4n plus 1 that time it is touching d. So let this be d and it is touching at 1, 4n plus 1. Then is 6, 6 it is taking b its schedule is 4n plus 2, so at 2 which means 4n plus 2 I am dropping the word 4n because it was getting clumsy, not otherwise. And then 7, 7 is c, input is c, But schedule is 3 so at schedule 3 it will touch this. Now start with Df 1 to 2, 1 to 2 means what is 1 adder, what is 2 adder, so adder to adder, this adder to adder, adder 2 it gives 1 input from adder itself 1 to 2 after one delay and what is 2? 4 is 2, 4 to 2.

Again 4 is an adder, so from adder itself it gets the input, other input (()) (23:34) is a 2 input with 0 delay. So with 0 delay I draw here, 0 delay. At what time will it take it schedule of 2 that is 1 so at cycle 1 it will take this. Switch will move here because that is schedule of 2, at schedule of 2 this is coming, this coming for what, this path 4 to 2 that is adder to adder with 0 delay adder output to adder input is 0 delay. So this will act, the act for the job up, act for the node 2, node 2 schedule is 1, So at schedule 1 it will take that input, what input from adder to adder with 0 delay. So, whatever was calculating, whatever came here that will come here without delay but that will actually was processed here at the previous cycle at schedule 0.

Schedule 0 is the schedule of 4 that is this adder actually took up the job of this adder for adder unit. This took of the job of adder 4 at schedule 0, I did the, calculated the output here. After 1 cycle of delay at schedule 1 this is coming down this path, this is coming down this path and available here at schedule 1, schedule1 is the schedule of 2. So that is why whatever 4 does that comes to the input of 2 at schedule1 because 4 has schedule 0, that time it took whatever with the input does the job, did the job got the output but output is delayed by 1 cycle. So at cycle 1 only output of 4 came over here. The other input of 2 is 1 to 2, the adder to adder, so adder output with one delay that means this I have to put to a delay and bring back here. At what time? Again at the schedule of 2, schedule of 2 is 1.

Similarly, start at 1, 1 has got 1 input in, another is coming from 3 to 1, in is coming when I have got the, in is the input to this node 1 that is when the hardware unit is working for node 1. When is it, at 4n plus 3. That time 1 input will be in. So 1 input of the adder will be in and it will touch it at 3, at 4n plus 3 because that takes this will in start working for node 1. So it will be touching the in at schedule 3, schedule 3 is the schedule of adder 1 and what is the other input. Other input comes from 3 to 1, 3 to 1 because (()) (26:50) was 0 so 0 delay. So again this will be touched, this will be touched at what, at what time, at the schedule of 1 at 3.

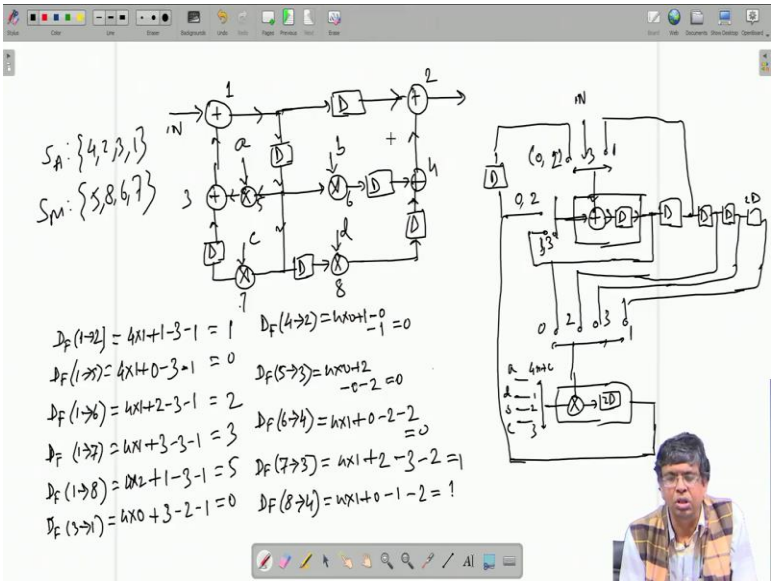
So what is happening actually at 3 it is coming but it was calculated by this node at schedule 2, schedule 2 means which adder has schedule 2, adder 3, so adder 3 calculated that at schedule 2 then at schedule 3 it brought out, 3 is the schedule of 1, 4n plus 3, that is why no delay required. It came here so 1, 3. So likewise we can go on filling. We have got 1 done, 2 done, 3, 3 comes from two multipliers, 5 and 7. But multiplier has only 1 output, 5 and 7. So multiplier output let us say 5 to 3, no delay. So no delay will go here, 5 to 3 what is the schedule of 3? 2, so at cycle 2 this will come here, 3 what is the input 5 to 3 but there is another input from 7, 7 to 3 through a delay 7 to 3 one delay that means other input this side will come after one delay from the multiplier.

At what point at 2 because 3 schedule is 2 so at 2 it will take from this without delay that is 5 to 3 and again multiplier output through one delay (()) (28:51). So both of schedule 2 because schedule 2 is schedule of 3, So 5 to 3, 0 delay this multiplier output 0 delay to 1 input of the adder and we touched at schedule 2, cycle 2 it means while it is working for adder 3. And the other one corresponding to 7 to 3 means after one delay that goes to the other input of the adder

again schedule should be 2 because that is what it should touch it when it wants to work for adder 3.

That was adder 3. Now adder 4, adder 4 also takes from multipliers 6 and 8, 6 to 4, 0 delay. So multiplier output comes here 6 to 4 what is the schedule of 4, 0 so it will be touched at 0. And 8 to 4, after one delay so after one delay, multiplier output one delay, it will be touched at what again schedule of 4 because I am concentrating on adder 4, so adder 4 has schedule 0. So other input after 1 delay this will be touched at 0 here. Alright, so 1, 2, 3, 4 done. Now comes multipliers, multiplier 5 comes from 1, there is adder, one delay and multiplier. So 1 to 5, 1 to 5, 0 delay that means multiplier output.

(Refer Slide Time: 30:38)



Let me do one thing instead of showing them here let me interchange the input for convenience of drawing. Let me take this to be the built-in constant side. So 4n plus 0, 1 there is 4n plus 1, 2, 3, a, d, b, c. Now as I was doing 5, I am focusing on 5, 5 schedule is 0 and 5 is taking input only from one adder so 1 to 5. So adder output after 0 delay. So after 0 delay comes directly here and I will be touching it at the schedule of 5, schedule of 5 is 0 so 0. Then adders, multiplier 6, multiplier 6, its schedule is 2 and that time I have got, you see 6 it is also comes from 1, 1 to 6. So 1 to 6, 2 delay.

So I have one delay, I need one more delay, the multiplier 6. It will come here at schedule 2, at schedule 2 is a schedule of 6. It will take the job of multiplier 6. Then next multiplayer is 7, 7

schedule is 3, 4n plus 3 and 7 takes input again only from 1, 1 to 7, 1 to 7 I have got 3 delays. So I have got 2 so I need another delay. This time it is working for 7, 7 schedule is 3 so 3 and lastly multiplier 8, its schedule is 1 and 8 takes again input only from 1, so 1 to 8, 5 delays. So 3 already there so 2 more, 2 D it will come here and it is for node 8, multiplier 8, 8 schedule is 1. So it will touch it at 1. This is the folded circuit.

Now here before I conclude you see none of the folded delay just turned out to be negative. But you can also see that it is not the conventional direct from 2 structure because direct from 2 structure, there would have been a delay here, no delay here, no delay here, no delay here, no delay here. It has been changed, actually it has been retimed. And we have got a structure for there is no negative edge. We will next show that by retiming you can come out of negativity. That is very interesting that we will show in the next class. Thank you very much.