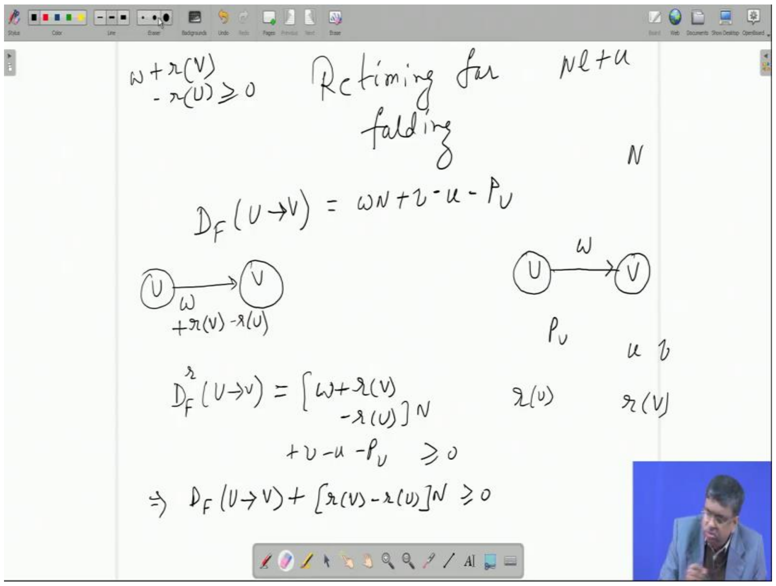
VLSI Signal Processing
Professor Mrityunjoy Chakraborty
Department of Electronics and Electrical Communication Engineering
Indian Institute of Technology Kharagpur
Retiming for Folding
Lecture 23

(Refer Slide Time: 0:28)



Retiming for folding

$w + r(V) - r(U) \geq 0$

$Nl + u$

N

$D_F(U \to V) = wN + v - u - P_U$

$D_F^r(U \to V) = [w + r(V) - r(U)]N + v - u - P_U \geq 0$

$\Rightarrow D_F(U \to V) + [r(V) - r(U)]N \geq 0$

So we are doing folding. Today I will be considering retiming for folding. We know that this formula DF U comma V was WN plus schedule v minus schedule u minus Pu. What was the context? In the DFG there is a node U, just a minute w number of delays, Pu is the number of pipeline latches in node U, these are all known. So I am just briefly recalling, w is the original number of delays, folding factor was N that is N nodes of the type u are backed to one single hardware unit of type Hu and that is pipeline by Pu number of delays, small u is the schedule of particular node capital U, small v is a schedule of particular node capital V. Then this particular edge is implemented in the hardware where one side is one hardware unit of Hu type, another side is Hv type. So the delay between Hu and Hv will be this much.

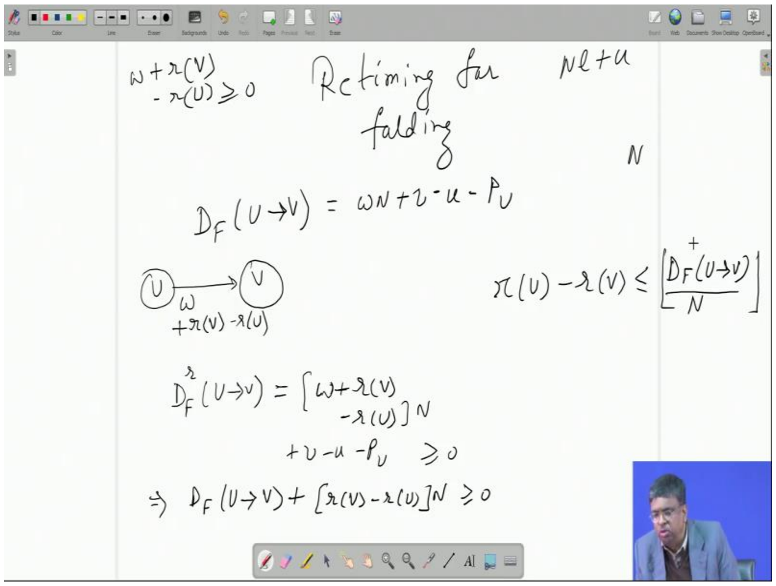
This is what we derive, can be used in one of the folded circuits earlier, assuming that schedule the hardware unit v works for the node v at a schedule v and hardware unit Hu works for the node u at a schedule u. we have got Nl for the lth word input what we have got Nl plus u, u could be 0,1 up to n minus 1. So, it is Nl plus u hardware unit Hu works for node u. Similarly, when it is Nl plus v hardware unit Hv works for node v. And the delay in between in them should be this much, this is what I did.

But the here the problem is there are minus signs and I told you that time that there is a possibility that sometimes because of your choice of vu, Pu and also because of the values you have for w and N this could be negative. One way to get out of the negativity is to first retime the DFG by assigning retiming values to various nodes and then do this folding. So suppose, I do retiming of this. So, I use r of U, r of U as a retiming value for node u, r of V as a retiming value for node v. Then after retiming we have got the delay changes from w to w plus r of V minus r of U. Of course, you should choose this so that feasibility is satisfied, feasibility and equality that is w plus r of V minus r of U that should be greater than or equal to 0.

Alright, that should be greater than equal to 0, w plus r of V minus r of U, that is one set so that you have studied in the case of retiming. That you assig retiming values, so that these inequalities satisfied, so nobody after retiming transfer to be negative alright. But now if I fold it, there is after the delay is change from w to this much, if I now fold it. Here in the formula only w will be changed to this value, other figures do not change, they remain same. So, if I call it DF r superscript, r for retiming that is after retiming then this delay expression will be w plus r of V minus r of U these are new delay that times n plus remaining things as before v minus u minus Pu and now I want this to be always positive, not negative.

This I can ensure by choosing rV, rU, they are my handles, I can choose them. So, using my schedules are, we are giving that the negative value here now they will not. How? Because if you simplify it, you take w into N plus v minus u minus Pu which is same as original DF U comma V. So this is equal to DF, U to V plus remaining portion is rV minus rU times N, alright rV minus rU times N. And this should be greater than equal to 0.

(Refer Slide Time: 5:52)



$\omega + r(V) - r(U) \geq 0$

Retiming for folding

$Nl + u$

N

$D_F(U \to V) = \omega N + v - u - P_U$

$$\begin{tikzcd} U \arrow[r, "\omega + r(V) - r(U)"'] & V \end{tikzcd}$$

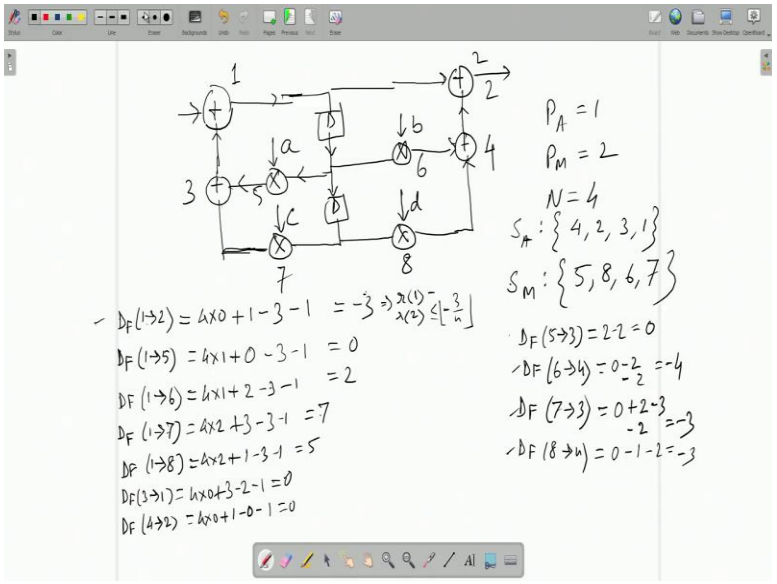
$r(U) - r(V) \leq \left\lfloor \frac{D_F(U \to V)}{N} \right\rfloor$

$D_F^r(U \to V) = [\omega + r(V) - r(U)]N + v - u - P_U \geq 0$

$\Rightarrow D_F(U \to V) + [r(V) - r(U)]N \geq 0$

So, if you take rV minus rU N on the other side, these give rise to another inequality, rU minus rV into N less than or equal to, into n I take it to this side DF U to V is originally before retiming whatever be the expression that you divide by N. Now rU is integer, rV is integer so difference must be integer. So I will take the integer value of this division that is if you divide take the quotient that is called floor. So rU minus rV this should be less than equal to this. So this is another set of inequality involving rU and rV apart from this. Other one was feasibility equality and now you have got another set of inequalities. So you have got a larger set of inequalities now if you solve them, you get appropriate values of rU, rV which you can use so that both feasibility is satisfied and none of the delay after folding like this, you know turns out to be negative because that is how this inequality has been obtained.

(Refer Slide Time: 7:16)



$P_A = 1$

$P_M = 2$

$N = 4$

$S_A : \{4, 2, 3, 1\}$

$S_M : \{5, 8, 6, 7\}$

$D_F(1 \to 2) = 4 \times 0 + 1 - 3 - 1 = -3 \Rightarrow \begin{matrix} x(1) - \\ x(2) \end{matrix} \le \left\lfloor -\frac{3}{4} \right\rfloor$

$D_F(1 \to 5) = 4 \times 1 + 0 - 3 - 1 = 0$

$D_F(1 \to 6) = 4 \times 1 + 2 - 3 - 1 = 2$

$D_F(1 \to 7) = 4 \times 2 + 3 - 3 - 1 = 7$

$D_F(1 \to 8) = 4 \times 2 + 1 - 3 - 1 = 5$

$D_F(3 \to 1) = 4 \times 0 + 3 - 2 - 1 = 0$

$D_F(4 \to 2) = 4 \times 0 + 1 - 0 - 1 = 0$

$D_F(5 \to 3) = 2 - 2 = 0$

$D_F(6 \to 4) = 0 - 2 - 2 = -4$

$D_F(7 \to 3) = 0 + 2 - 3 - 2 = -3$

$D_F(8 \to 4) = 0 - 1 - 2 = -3$

Alright, I take an example now take another filter, IIR filter and this is the conventional IIR filter you are familiar with. 1 this is 2, 3, 4, 5, 6, 7, 8. These are the conventional digital filter. All of us know direct form to digital filter, pipeline latches PA equal to 1 as before, multiplier requires 2, 2 latches, N is 4 that is 4 adders to be replaced by one hardware adder unit. And 4 multipliers to replace by 1 hardware multiplier unit very much like the example one which we considered earlier. And schedules also are same as which we consider last time, SA folding set for adder 4, 2, 3, 1, SB, sorry S multiplier I am sorry, SM.

Now like the previous example we calculate the delays after folding, so 1 to 5, DF 1 to 5 will be one delay between 1 to 5 so 4 into 1 plus v, v is the destination is node 5 or maybe I start with 1 to 2, that is better let us start with 1 to 2. Adder to adder then we will consider at to volume, 1 to 2 no delay in this edge so 4 into 0 plus v that is the schedule of node 2, schedule of node 2 is say 0, 1, 2, 3 that is 1 minus schedule of node 1, node 1 is 3 so 3 minus pipeline latch for adder which is 1. So this is negative, the first one is turning out to be negative, minus 3. We want every edge delay to be positive but here I am getting negative, minus 3.

Fine fair enough let us still proceed then 1 to 5. Here I have got one delay, so 4 into 1 plus 5 means schedule 0. This part remains same because that delays to adder 1 so its schedule the number of pipeline latches do not change minus 3 minus 1 which is 0, no problem. Then DF 1 to 6, DF 1 to 6 again one delay so 4 into 1, 6 as a schedule 2 (()) (11:16) as before because its adder

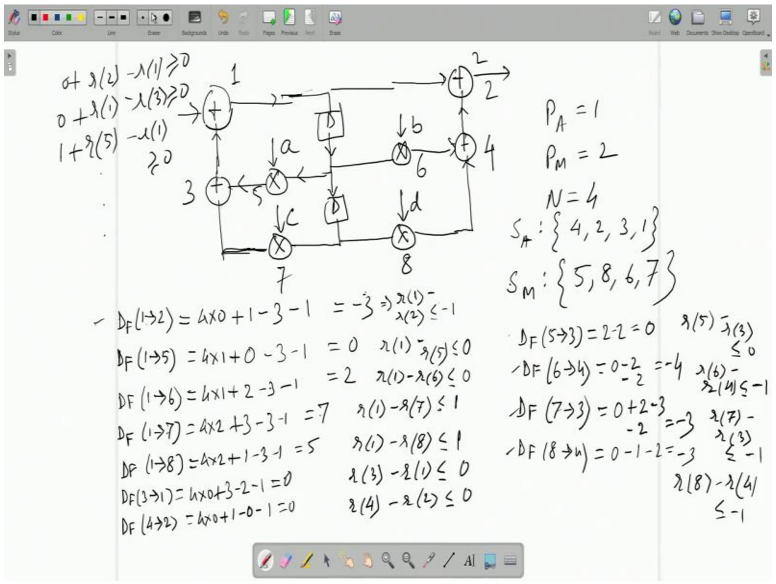
1. So 4 into 6, 3 to this is 2, 1 to 6 is 2. Then DF 1 to 7 two delays so 4 into 2 plus schedule of 7, which is 3 minus this part remain same so this is 7 positive, no problem. Then DF 1 to 8, 1 to 8 also two delays 4 into 2 schedule of 8 is 1 this part remains as before, so it is 5.

So I am done with 1, say DF 3 to 1 no delay so 4 into 0, 1 has a schedule 3, minus 3, 3 has a schedule 2 and 3 also is an adder so pipeline latch is 1. So this is 0 no problem. Then DF 4 to 2 that I am still considering adder to adder no delay, so 4 into 0 plus 2 adder 2 has schedule 1 minus 4 schedule of 4, schedule of 4 is 0, at still adder, so pipeline latch is 1 so it is having 0. Then comes other one, multipliers 5 to 3, 5 to 3 I am directly working out 5, no delay so 4 into 0 plus 3 is a destination it has a schedule 2 minus 5 is a source schedule 0. So 2 minus 02 minus the number of pipeline latches for multiplier which is 2, Pm is 2, so 2 minus 2, which is 2 minus 2 which is 0.

Similarly, DF 6 to 4, DF 6 to 4, 4 is the destination which an adder schedule 0, 6 schedule 2 and no delay in this path, 0 minus 2 minus number of pipeline latches in multiplier 6 which is 2 so again I am getting a minus quantity. Then DF 7 to 3, 4 into 0 plus 3 has a schedule 2 minus schedule of 7 is 3 minus 2 because pipeline latch is 2 here. So again I am getting a negative quantity minus 3 no delay, so 4 into 0 plus schedule of 4. So the adder 4 is 0, so 0 minus schedule of 8 here is 1 so minus 1 and again multiplier is a source so its number of pipeline latch is 2 so minus 3. So these are the things this fellow, this fellow, this fellow and here one fellow they are the troublemakers, they are negative.

So I will be retiming, there is I will be assigning retiming values r of 1 to node 1, r of 2 to node 2, r of 3 to node 3 like that. And then I will get some inequalities and in addition to the feasibility inequalities, what are the new inequalities we have already seen the formula rU minus rV so if you consider 1 to 2, this will give rise to r of 1 minus r of 2, r of 1 minus r of 2, U to V, r of U minus r of V, this should be less than equal to original DF divided by capital N that is 4 and it is there is minus 3 by 4 its quotient, minus 3 by 4, you have to go to the integer, which is just below this. So minus 3 by 4, the integer below this is minus 1 because it should be less than equal to minus 3 by 4. Ideally it should be less than equal to minus 3 by 4, but r1 is integer, r2 is integer, difference is integer so if the integer is to be less than equal to minus 3 by 4, so it has to be less than or equal to minus 1, because minus 3 by 4 is not an integer.

(Refer Slide Time: 16:40)



$0 + r(2) - r(1) \geq 0$

$0 + r(1) - r(3) \geq 0$

$1 + r(5) - r(1) \geq 0$

$P_A = 1$

$P_M = 2$

$N = 4$

$S_A : \{4, 2, 3, 1\}$

$S_M : \{5, 8, 6, 7\}$

$D_F(1 \to 2) = 4 \times 0 + 1 - 3 - 1 = -3 \Rightarrow r(1) - r(2) \leq -1$

$D_F(1 \to 5) = 4 \times 1 + 0 - 3 - 1 = 0 \quad r(1) - r(5) \leq 0$

$D_F(1 \to 6) = 4 \times 1 + 2 - 3 - 1 = 2 \quad r(1) - r(6) \leq 0$

$D_F(1 \to 7) = 4 \times 2 + 3 - 3 - 1 = 7 \quad r(1) - r(7) \leq 1$

$D_F(1 \to 8) = 4 \times 2 + 1 - 3 - 1 = 5 \quad r(1) - r(8) \leq 1$

$D_F(3 \to 1) = 4 \times 0 + 3 - 2 - 1 = 0 \quad r(3) - r(1) \leq 0$

$D_F(4 \to 2) = 4 \times 0 + 1 - 0 - 1 = 0 \quad r(4) - r(2) \leq 0$

$D_F(5 \to 3) = 2 - 2 = 0 \quad r(5) - r(3) \leq 0$

$D_F(6 \to 4) = 0 - 2 - 2 = -4 \quad r(6) - r(4) \leq -1$

$D_F(7 \to 3) = 0 + 2 - 3 - 2 = -3 \quad r(7) - r(3) \leq -1$

$D_F(8 \to 4) = 0 - 1 - 2 = -3 \quad r(8) - r(4) \leq -1$

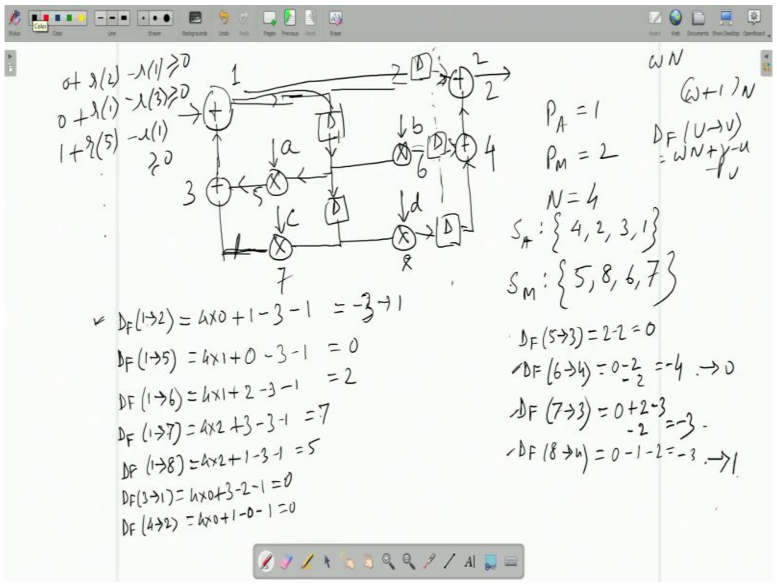
So therefore, you can write it this way minus 1, sorry. Similarly, r1 minus r5, r of 1 minus r of 5 this will less than equal to 0 by 4 and the integer below that less than equal to, So 0 by 4 integer itself is 0 which is itself is an integer which is 0, so do not need to go below it because it is not a fraction. So this is another inequality then r1 minus r6 less than equal to 2 by 4 that is 1 by 2, we have to go one step below one step below the next integer immediate below 1 by 2 that is 0 2 by 4 that is half so r1 minus r6 should be less than equal to half.

But this integer, this integer difference is an integer so integer must be less than equal to half, it must be less than equal to 0. Because this side is integer right side also be integer that is why I am taking the floor. There is an next integer on the lower side., then r of 1 minus r of 7 less than equal to 7 by 4, 7 by 4 is 1 plus 3 by 4. So integer just below this is 1 so 1. Similarly, r of 1 minus r of 8 less than equal to 5 by 4, which is 1 plus 1 by 4 integer below is 1, r of 3 less than equal to 0, r of 4 0. Alright, around this side r of 5 minus r of 3 less than equal to 0 of course 0 by 4 which is integer 0, so it remains 0 here. Then r of 6 minus r of 4 less than equal to minus 4 by 4, which is minus 1, but there is an integer, no need to go below that.

So minus 1, then r of 7 minus r of 3 less than equal to minus 3 by 4 which is minus 3 by 4 is a fraction so integer below that is minus 1 so minus 1 and same here r of 8 minus r of 4, here also minus 3, so it will be minus 1. So you get a 6 set of inequalities and you have got the feasibility in equalities. Feasibility in equalities very simple for such between 1 and 2, it should have original w which is 0 plus r of 2 so 0 plus r of 2 minus r of 1 that should be greater than equal to 0. So in between 3 and 1, 0 delay plus r of 1 then there is a destination minus r of 3 greater than equal to 0 so and so forth 1 to 5, 1 to 5 is 0 plus, not 0, here is one delay. So, so 1 plus r of 5 minus r of 1 greater than equal to 0 dot dot dot dot.

We know how to do that. So you have a set of inequalities called feasibility in equalities, which come from retiming and then you have got a set of inequalities which come from a folding consideration so you have got larger set of inequalities hoping that this has a solution space, if you give to a computer package it can give a solution sets. And using any such solution set you can use, you can retime it the delays will be distributed now. Because if you have r1 some value and r5 some value so these will get changed to 1 plus r of 5 minus r of 1 and like that. So you change the delays, it will be coming out of this negative problem. This a general way of doing but a practical way of doing applicable to this (()) (21:17) circuits, is often obtained by cut set, cuts set retiming. Let me explain them, explain that.

(Refer Slide Time: 21:27)


0 + r(2) − r(1) ≥ 0
0 + r(1) − r(3) ≥ 0
1 + r(5) − r(1) ≥ 0
P_A = 1
P_M = 2
N = 4
S_A : {4, 2, 3, 1}
S_M : {5, 8, 6, 7}
WN
(W+1)N
D_F(U→V) = WN + P_U − u + v
D_F(1→2) = 4×0 + 1 − 3 − 1 = −3 → 1
D_F(1→5) = 4×1 + 0 − 3 − 1 = 0
D_F(1→6) = 4×1 + 2 − 3 − 1 = 2
D_F(1→7) = 4×2 + 3 − 3 − 1 = 7
D_F(1→8) = 4×2 + 1 − 3 − 1 = 5
D_F(3→1) = 4×0 + 3 − 2 − 1 = 0
D_F(4→2) = 4×0 + 1 − 0 − 1 = 0
D_F(5→3) = 2 − 2 = 0
D_F(6→4) = 0 − 2 − 2 = −4 → 0
D_F(7→3) = 0 + 2 − 3 − 2 = −3
D_F(8→4) = 0 − 1 − 2 = −3 → 1

Now consider this, I have got these 4 fellows who have work creating the trouble, they are negative. Now you understand that, if any edge I increase the delay by 1. So, if it was originally w then with the folded figure you know, you have WN factor coming. If w goes up in any edge by 1, w plus 1. So it becomes w plus 1 into N. So delay increases by N. You understand that what I am talking of. In the folded delay formula DF we have WN plus small v minus vu minus pu if w is increased original edge by just 1 delay in this formula it increases by capital N.

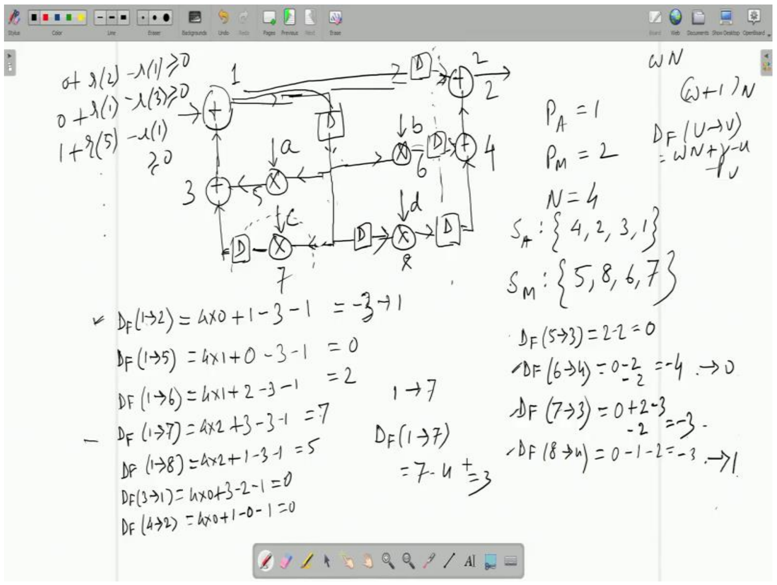
So therefore consider 1 to 2, 1 to 2 there is no delay, the edge is like this is 1 edge, this is 1 to 2 direct edge and if in this age, I have 1, I now introduce suppose 1 delay by hook or crook. Then instead of 4 into 0, it will be 4 into 0 plus 1. So 4 factor, an additional 4 will be added so 4 minus 3 it will be plus 1. Similarly, here also minus 4, 6 to 4 there is no delay, if I can introduce a delay here like here, here if I can introduce a delay here. If I can introduce a delay here, then what happens earlier I had 0 now it will be, that is 0 means 4 into 0 now it is 4 into 0 plus 1. If I have delay 1 here, remaining things are same. So plus 4 factor like the plus N factor, plus 4 factor would come up here that will get added with minus 4 it will become 0 it will become 0.

This will become 1. Similarly, 7 into 3, minus 3, so 7 into 3 if here I have one delay, if here I have one delay then in this calculation instead of 4 into 0 it would be 4 into 0 plus 1 that is 4 into 1. So 4 factor would get added with minus 3. So it will become plus 1 and last one 8 to 4, 8 to 4 again no delay if I have a delay here, there 4 will get added 4 minus 3 it will become plus 1. But I cannot arbitrarily add delays but if I add delays that amount to cut set retiming I am done because cut set retiming is valid retiming. Now you see, see the mark I want a delay here, delay here, delay here.

Clearly if I have, If I just remove these edges this one, this one, this one they form a cuts set. Because the circuit is divided into one side this side adder 2, adder 4 at this edge and remaining part here. So they form a valid cut set and all are in the forward direction, left to right, left to right. So I can happily add a delay here so I can put a delay here. So fine, so here I just added a delay from outside there was no delay, they are not present here but it is forward path pipelining, forward path retiming I just added delay, they form a cut set these 3 edges, if that is if you cut them, the hint that the end is cut into 2 to form a valid cut set only one direction.

So I happily add a constant number of delay 1 delay in all of them. So this becomes positive minus 4 to it becomes 0 from minus 8 it becomes 1, from minus 8 it becomes 1 but other columns are this side 7 to 3, sorry not here. I am really sorry, not here, 1 to 2, these three, 1 to 2, 1 to 2, 6 to 4, 8 to 4 but now I am left with 7 to 3 that is another direction that is in a loop. Like if you go through this way the node1then this 2 delay then node 7 then node 3 and that. In a loop, I cannot bring in delay, new delay from outside that violates retiming as we have shown, so we have to use existing delays of the loop for doing retiming, I kind of like unlike here where these delays were not at all ingesting I brought in from outside world. Here I cannot do that, but I must have a delay here.

(Refer Slide Time: 26:47)



0 + λ(2) − λ(1) ≥ 0
0 + λ(1) − λ(3) ≥ 0
1 + λ(5) − λ(1) ≥ 0
P_A = 1
P_M = 2
N = 4
S_A : { 4, 2, 3, 1 }
S_M : { 5, 8, 6, 7 }
wN
(w+1)N
D_F (U→V) = wN + y − u − P_V
D_F (1→2) = 4×0 + 1 − 3 − 1 = −3 → 1
D_F (1→5) = 4×1 + 0 − 3 − 1 = 0
D_F (1→6) = 4×1 + 2 − 3 − 1 = 2
D_F (1→7) = 4×2 + 3 − 3 − 1 = 7
D_F (1→8) = 4×2 + 1 − 3 − 1 = 5
D_F (3→1) = 4×0 + 3 − 2 − 1 = 0
D_F (4→2) = 4×0 + 1 − 0 − 1 = 0
1 → 7
D_F (1→7) = 7 − 4 = 3
D_F (5→3) = 2 − 2 = 0
D_F (6→4) = 0 − 2 − 2 = −4 → 0
D_F (7→3) = 0 + 2 − 3 − 2 = −3
D_F (8→4) = 0 − 1 − 2 = −3 → 1

So, what I can do is this, this is 1 edge you can view it this way. May be you can just put a dummy node here dummy, which is accepting these and splitting it in various directions. Through this dummy node there is 1 delay going in this direction and another delay going in this direction because that this delay was common. I have just split. Now, see what edges I am cutting, I am cutting this edge and this edge. So there form a valid cut set, because if I remove this and this, these nodes is a single turn left out and remaining part of the circuit another part. So the entire DFG circuit is cut into exactly two mutually separate DFG. So one is the single node 7, other is the remaining part of the DFG.

But one is incoming direction that is edge is coming from other half of the DFG to node 7 another is going out of node 7 to the other part of the DFG. In one incoming is there is a delay this I can cut, so I can bring back the delay on the side. In one direction, I will remove and add the same delay in other direction, there is the formula., now we can merge the two lines. Take the delay here, this one from delay has taken out and here the other 8, which has delay that delay I am putting here, this is a circuit we considered the other day. But only thing you should check is this in the forward path pipelining there was no problem I brought an extra delay, but in the loop wherever I moved the delay earlier there was a delay here, here I moved it here.

So in this movement because some delay which was existing already in the part of the in one of the edges that have moved to some other edge, so maybe one of the edges now getting less delay like this edge, we can remove this dummy node now, this edge earlier had 2 delays which edge 1 to 7, 1 to 8 earlier had two delays now also two delays no problem, but 1 to 7 I had one delay here so, 1 to 7 had two delays. One of them now has moved up. So 1 to 7, here one delay has come down, which means from DF 1 to 7 whatever will be the value, 4 will get subtracted earlier I had 4 into 2 like here 4 into 2, now it will be 4 into 1. So, 4 will be subtracted, question needs to be checked is this does it lead to new negative edge? Here fortunately not because it will be now 7 minus 4 which is still positive 3.

But if it so happens that after in subtract 4 but the original value we get negative then you cannot do anything. So, this needs to be checked after you complete retiming and all you have to check whenever delay has moved from one edge to another edge. So some edge is getting extra delay some edge is getting less delay than earlier, whichever edge losses some delay, you have to recheck whether the now the new value of this folded delay is still positive or its negative. If positive you are done. Alright, this is a practical way of doing folding, I mean with retiming that is using cut set but remember you have to do this cross checking at the end also. So, that is all for this. Thank you very much.