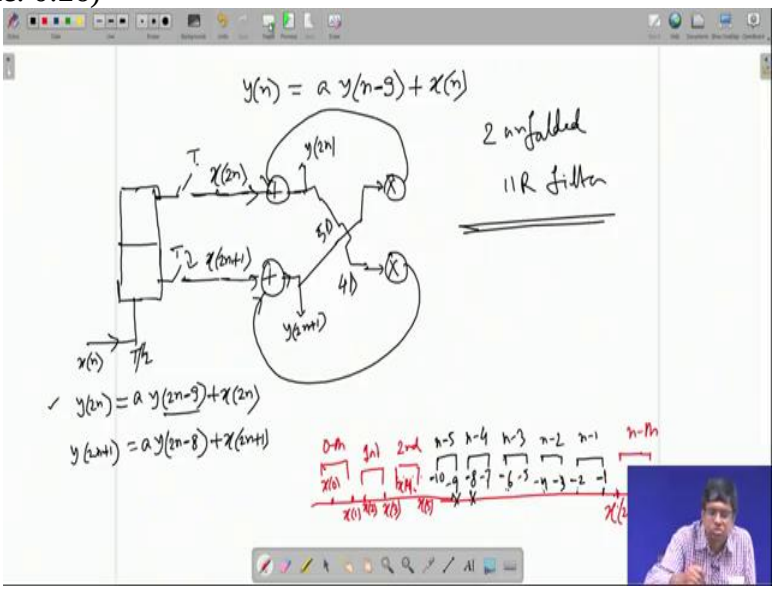
VLSI Signal Processing
Professor Mrityunjoy Chakraborty
Department of Electronics and Electrical Communication Engineering
Indian Institute of Technology, Kharagpur
Lecture 14: Basic Unfolding Relation

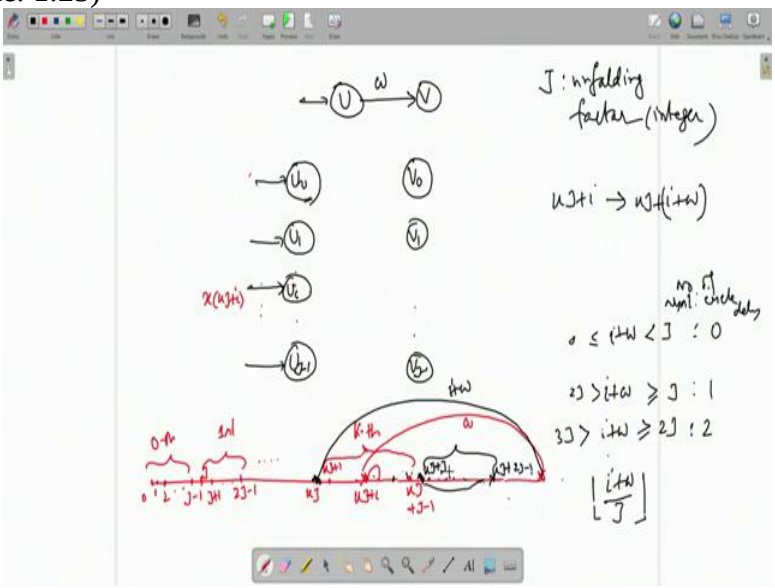
(Refer Slide Time: 0:26)



Okay, so we completed this, there is a 2 unfolded IIR filter equation given here. Now as I told you, this is just as an example, it is fine but then I need to show you how to carry out such parallelization, there is unfolding by maybe a factor of J, J is an integer, in a general DFG. So that is what some general results I derived now.

A DFG means what? Typically a DFG means there will be a node and edge coming out from it and hitting another node, the edge may have some delay or may not have some delay and the entire DFG will be repetition of this, one node to another node, one edge with a delay, from that node to another node, some edge with or without a delay so on and so forth, entire DFG you can break, I mean you can just reverse, cascade off or repetition of such structure.

(Refer Slide Time: 1:25)



One node U, edge may come up, come out of it with w number of delay and V, and again from V there will be another edge going to another node with w prime number of delay and so on and so forth, the entire DFG will build up like this, but this is a basic sector of a DFG. We will unfold, that is parallelize the DFG by sector by sector.

That means we will just take this sector parallelized and then next sector parallelized. So entire DFG will be parallelized or parallelizing factor which is unfolding factor is J, which is an integer, unfolding factor this topic comes under Unfolding, the title Unfolding. This is an integer as I told you. So, either it is U was taking input directly, so input will be put in the buffer of J chambers, U is the process it will be copied J times, there will be switches as I have shown. They will be taking data from the J's chambers and it will give to the parallel copies of U, but in a more general sense maybe this is not taking input from a directly input source, there may be another node or some other process.

So, another process that is also parallelized that is giving its output parallelly. So, instead of one single line or one process, this process actually will be copied, parallelized U0, earlier there was one line and one process. Now there will be J copies maybe U0, U1…Ug minus1, identical parallel, but I'm leaving them as U0, U1…Ug minus 1. Earlier I had one line, now, I have got parallel lines.

Similarly, V was taking data from one line now, there will be parallel processes V0, V1…Vg minus 1 and they will take data separately, one may take data from one of them, another may take data from another, like the parallelly.

We will find out if I take the output of U0, which one of these processes V0, V1 or Vg minus 1 it will go to. After all, output of U has to go to V, U type of processor must deliver to V type of processor. So, if it is U0, if it is the U type of processor it will give its output to one of the V type of processors but which I have to find out and in that edge what will be the delay? Because this input, this delay w is with respect to the original input clock, faster clock, okay original, but when you parallelize, when you parallelize, system will be working at a slow clock. So, it will not be input clock, this is the original DFG with respect to the original input and output rate; with that rate, there is w delay, w number of delay.

But when you are parallelizing, the system will be slow, input will be faster, okay, the system clock will be slow. So, system clock period will go up by a factor J or rate will go down by factor J. In terms of system clock, how many delay per edge here, in terms of the system clock, that is what we find out. And take any processor here, say U i, which of the V processors it will go to? And in that edge, how much, how many number of systems I can delay?

I repeat, this diagram is shown with respect to original DFG where input was coming at a faster rate, so, J times faster rate and then in terms of that clock I have got w number of delay here. But when you parallelize, system is slower, input is faster. So, system clock is slower, system clock's period is J times the original input clock, or speed is J times less. So, in terms of this system clock because they are working on a system clock, how many number of delays per edge between U type to V type processors? To answer this question, I will come back to the timing diagram.

If you start at say U0, 0 is data, the data at 0 is cycle here, then first cycle, second cycle…they are all parallel, data at 0, 1, 2,… up to J minus 1. Here they are coming serially, but because of the buffering, they are all parallel. And they appear in the system, system is here together and processed in one clock, that clock I am calling 0^{th} system clock. So this much, so system clock is slower. It takes J number of input clocks, if system clock period is capital T, input clock is T by J, as the w delays interrupt the input clock with period T by J, w number of input clock of period T by J where system clock is T, once a system clock which I call 0^{th} it carries the input data x of 0, x of 1, x of 2 up to x of J minus 1.

Originally, there were coming serially. Now, they are parallel, why parallel? Because of some buffering say, then is your fast system clock, it will start at J then J plus 1…J means J plus 0,

J plus 1, J plus 2 to J minus 1, so 2J minus 1, it is 0, 1, up to J minus 1, total J. Then J means J plus 0, then J plus 1, J plus 2…J minus 1, J plus J minus 1. So total 2J minus 1… So, x of 0, x of 1, x of 2 up to x of J minus 1, they are coming serially at a faster input clock of T by J period, they are now mid-parallel and they are coming into the system parallelly through those J parallel lines but at a period of capital T.

Then again x of J, x of J plus 1, x of J plus 2…x of 2J minus 1, they were earlier coming serially down a line one after another with a clock period of T by J. But now they are bit parallel, they are coming parallelly along these lines and they are entering the system at a period of T, so on and so forth.

In general, you have, I am drawing in a bigger way. So, this is one period looking bigger that it is just for explanation I am making it bigger so that you can see things clearly. Suppose I have reached the Kth, Kth period, so what will be the indices covered and what will be the input indices? See 0 to J minus 1, it is under 0; J to 2J minus 1, it is under 1; under 2 will be 2J to 3J minus 1, so under K will be KJ, then KJ plus 1….in general, there may be KJ plus I and KJ plus J minus 1. So, KJ means KJ plus 0, then KJ plus 1, then KJ plus 2, KJ plus 3, then KJ plus J minus 1, so total J. All this data which are coming serially originally at a period of T by J, they are now made parallel, parallelly available, they are entering the system and being processed, entering the system at a period of capital T or rate of 1 by T.

Out of which my convention is this, this data x of KJ, that is KJ plus 0 that will come to U0; KJ plus 0, U0; data KJ plus 1 it is coming parallelly, it will be given to U1. So KJ plus 1 going to U1, so, KJ plus I will go to UI, so x KJ plus I goes to UI alright, so on and so forth. x KJ plus J minus 1 will come to UJ minus 1, these are conventions. So, x of KJ here, x of KJ plus 1 here, x of KJ plus 2…x of KJ plus I, then x of KJ plus J minus1 here, all of them are parallel and coming at a period of capital T or 1 by T rate, alright.

Now they are processed by this, fine, but after getting processed each output has to be delayed by w number of input cycle, this cannot be violated, this is the original job, whatever I do for digital implementation parallelly or whatever that is my business. But original job cannot be violated, original job was that every time you process the original input at the original rate, after that the output comes at the U type of processor, you have to delay it by w number of original input cycles or the cycle period was T by J.

So, that means, this fellow, original case, this is generating the output, so, at KJ plus I suppose it is generating the output, suppose its processing time is 0 for the time being KJ plus I. So, after that KJ plus I, I have to go to the right by w number of input cycle. Input cycles are sorted, so I count like this 1, 2….okay. So, essentially, if I count KJ plus I plus 1, KJ plus I plus 2, KJ plus I plus 3 like that, I have to count w number of cycles. So, basically from this point, I will be jumping to the right by w number of input cycles. From here I will be jumping to the right, I will come somewhere here, w number of input cycle, that is 1, 2, 3, 4 like that, I get here.

That means KJ plus I, x KJ plus I comes here, processed by U I, this output has to be delayed by w number of input cycles, that original time scale it should be here.

So, in terms of the system clock, how many system clocks I have within this zone? So, that many system clock I have to hold this in some way, for that but many system clock I have to hold this output in some delay register because system is working with respect to the system clock not with respect to original clock. System clock is J times slower, period is J into T by J that is T, original is T by J, it is now J into T by JT or rate is slower, period is longer.

Alright. If I count I am here so, within this how many system clocks will be required to reach here? For that many systems clock I have to hold the output in delay, that is one thing but before I proceed further, but if it is KJ plus I this point is x KJ plus I plus w. Let it me, if it is KJ plus I, no need to put x, this is going to KJ plus I plus w, alright. Then I can write this way, it as though KJ plus I plus w. That is, if I start instead of from KJ plus I if I start here at KJ, then I have to jump to the right by I plus w.

If I start at KJ, I am jumping to the right by I plus w, so, instead of counting at KJ plus I, I start counting from KJ and jump to the right by I plus w. Within I plus W how many blocks of J data? Well, each block of J data means one system cycle, another block of J data means, J input data means another system cycle.

So, how many blocks I have? That is what I am going to find out, so, that many system cycles will be required to hold this output, before I can reach this point. Alright, let me explain further. So suppose I start here, but w small, suppose w is less than J so, that means w can be maximally up to J minus 1, maximally. So, from KJ it will be I plus w, so, this point it will be…suppose it start here, if I plus w, sorry if I plus w is suppose less than J, I plus w is suppose less than J, now w I plus w that is maximally I plus w J minus 1.

So, either it will be from here it will go up to KJ plus J minus 1 or will be before this but they are all under the same system clock Kth system clock. So, KJ will remain under the same umbrella Kth umbrella, it will just move from one point to another point. So, KJ plus I will go to some other point but it will not require any system cycle delay because it is lying within the same Kth umbrella. Okay, KJ, KJ plus 1, KJ plus 2 up to maybe KJ plus I which is either here or before, this so, I do not need to store it in a system clock because they are available together parallelly, so, simultaneously.

But KJ means KJ, okay, I have to find out what that point is, that is later. But suppose now, it is not less than J, it is greater than equal to J but less than 2J, that means, if you take the next block, next block starts at KJ plus J, it is KJ plus J minus 1 so, next block starts at KJ plus J and goes up to KJ plus J plus J minus 1, so, 2J minus 1. So if I plus w is greater than equal to J so, KJ you can start, minimum is J so, you can go here or here, here, here, but it is less than 2J, so it can be up 2J minus 1. So, you either jump to the next cycle, the starting line or next or next or next or up to here. So in this case you are in the next block. So that means you need one system cycle delay because all this data are parallelly available in Kth system cycle delay, now, they will be going to K plus 1th.

So, if I plus w is between, that is if I write I plus w, earlier it was less than J greater than equal to 0, I need 0 system cycle delay, number of system cycle delay, system cycle delay 0. Then if I plus w, this is I plus w, if you cannot see clearly let me, okay, if I plus w this I plus w is the factor by which you are jumping. If I plus w is now this so, you are jumping from what? I mean I plus w is greater than equal to J. So, KJ means KJ plus J because minimum is J so, either here or J plus 1, J plus 2 that is KJ plus J plus 1, KJ plus J plus 2…less than 2J means up to 2J minus 1. So, KJ plus 2J minus 1 so, under this condition if you jump by I plus w, you hit this block.

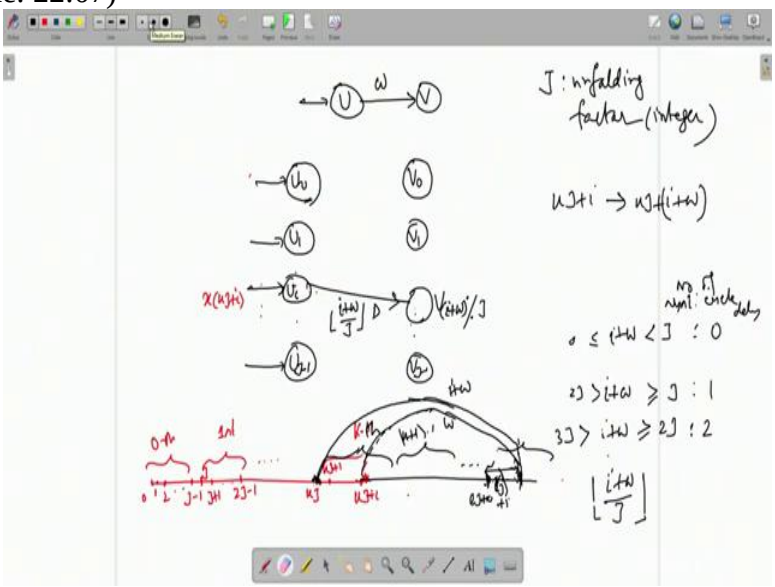
So, you need one system cycle delay because from Kth this is K plus 1th, then again if you by this way you can proceed. If it is I plus w is greater than equal to 2J, less than 3J, it will be the next block.

So, you need two system cycle delay….so, actually if I divide I plus w by J and find out the quotient, if I plus w is less than J, quotient is 0, I need 0 system cycle delay. If I plus w is just greater than equal to J but less than 2J, if I divide I plus w by J, quotient will be 1 and you see system cycle delay is 1. If I plus w is greater than equal to 2J, but less than 3J and if we

divide by J, quotient will be 2. Basically in one I plus w, how many blocks of J size are hidden, that is what we find out by making this division taking the quotient. The quotient is denoted by this floor, that many systems cycle delay is required.

So, KJ plus I this data will have to be delayed by I plus w by J quotient, that many system cycle delay you have to offer to the data at KJ plus I. There is the output coming out from UI because as I told you KJ plus 0 is at U0; KJ plus 1, U1; KJ plus I, UI, so, UI output has to be delayed by I plus w by J quotient, these are notations for quotient. That is fine, but now there is one more issue.

(Refer Slide Time: 22:07)



So you have got one block, this is Kth, this is K plus 1th….I plus w is hitting some other block….K plus 1, K plus 2. So if from K to K plus 1 if you go, you need one system cycle delay. From K to K plus 2 you go, you need two system second delay. Here you are from Kth you are going to this block. So within I plus w how many J? 1J, 2J, 3J how many J's? So we divide and take the quotient, that many system cycle delay is required, fine.

But once you hit this block, I mean you could either hit the starting point or next point or next point…or some point here, which point you hit that will be the remainder, what does the remainder indicate? If the remainder is zero, you hit this point, starting point; starting point means you have to give the job to V0 because starting point means this whatever may be the, you know, it may be some integer like LJ plus 0, then LJ….the starting point is always like that, 0 then J plus 0, 2J plus 0, KJ plus 0, so it will be some LJ plus 0.

If this goes here, that is, suppose this was hitting here starting right with LJ plus 0, actually I plus w or equivalently, actually you are delaying by w, I counted from KJ for my convenience but actually you are starting from KJ plus I going to w. If you hit here, that means this output has to be not only delayed by so much of system cycle, it has to be given to the processor V type of processor which is engaged for processing data at the starting index, that is V0, with V0 for LJ plus 0 or KJ plus 0, V1 for LJ plus 1 or KJ plus 1 like this point, like that.

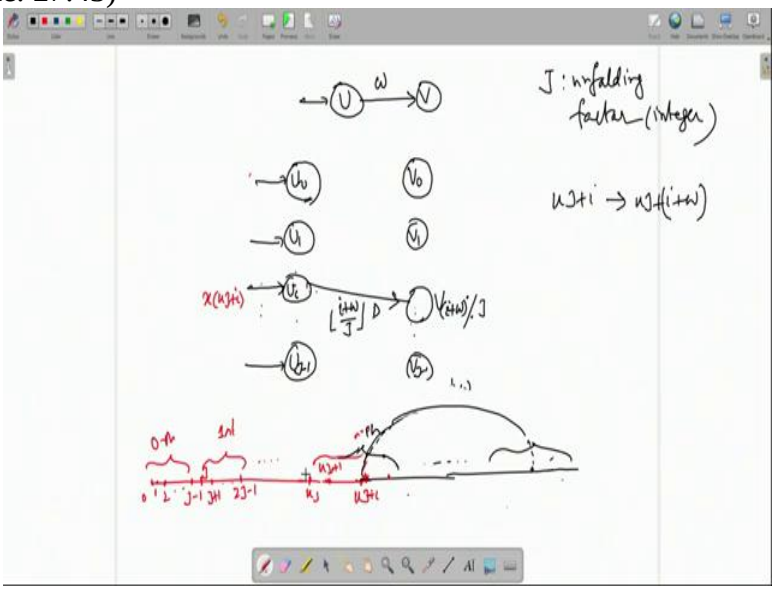
So, similarly by jumping, if by jumping over to the next point so, it is LJ plus 1, that means KJ plus I, actually this is the actual point, I count it from KJ for my convenience but actually from KJ plus I that is from here, you should delay it not only by this much amount I plus w by J quotient but you are hitting this point now, that is LJ plus 1. So with LJ plus 1 it should go to V1 because point number one, point number zero is here, point number one, one is dedicated, V1 is dedicated for those points, okay.

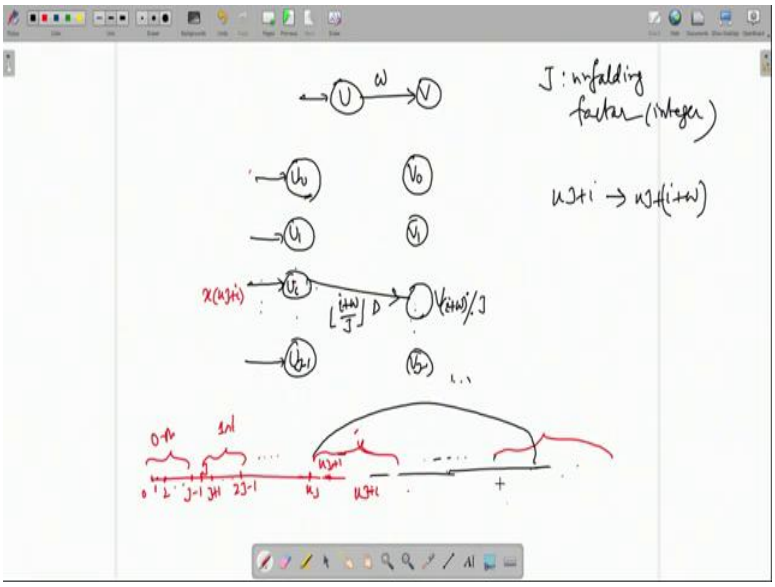
So, in general how much will this be? This is the remainder. In general, you are hitting somewhere here. How much is this part? This is 0, this is 1…how much is this part? Obviously, if you do this division, whatever remainder you get that is the remainder, so remainder will be your index, V of that index will be receiving output from UI.

So actually this is the formula we get, V of remainder is denoted like this, if I divide I plus w by J, sorry, this is V, I am writing V outside, I plus w, if you divide by J and take the remainder, it is denoted by this. So, this much is the remainder I plus w by J division, this much is the remainder. This remainder is the index that V0 for here, V1 for here, V of whatever it is so, I plus w percentage J that is this processor, that is taking data from UI, that is taking data from UI and there will be delay, system cycle delay of this much amount. If I divide I plus w by J take the quotient, that many systems cycle delay of period capital T, input original is T by J, original input to this parallel system coming at T by J, final output also out of the buffer will go out at T by J, system is working at a period T. But this is a general formula we have derived, this is very important.

We will then, in the next class we will try to understand the implication of this formula, what kind of structure it gives rise to, here only you can see one thing.

(Refer Slide Time: 27:43)

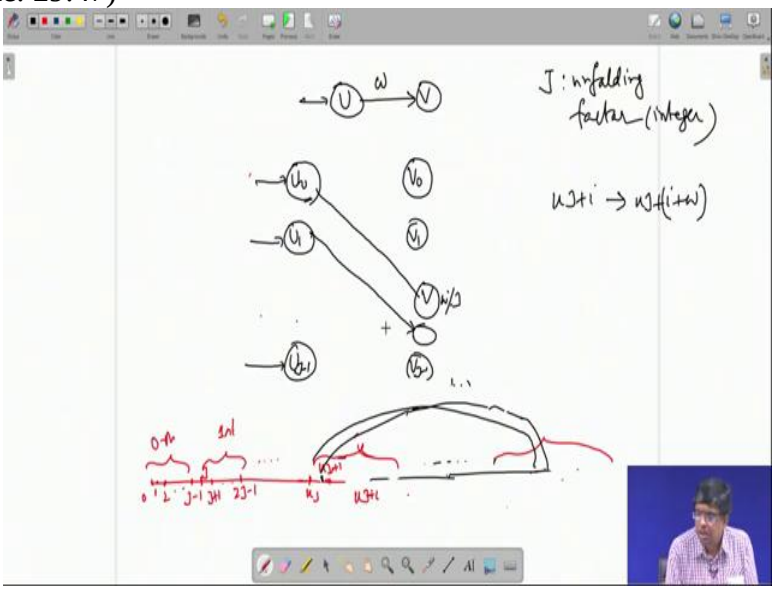




I can erase this part and I can make this drawing little fairer, no longer I need this counting from KJ, it is my block KJ, KJ plus 1, like this one block, I do not need this block, K plus 1th. From here you are jumping…You are hitting some block at some index here, from KJ plus I you are starting. So let us start at, instead of KJ plus I let us start at KJ plus 0. So, KJ plus 0 means data out of U0 and then I will make it KJ plus 1, KJ plus 2, like that.

We start at KJ plus 0, I is 0 so, you jump to some point here, you find out some, I mean KJ plus I, I is 0 here. So, I plus w, that is 0 plus w by J. So, just divide w by J, take the quotient, whatever that many system cycle delay and find out the remainder that will be your, that will give you the index of the processor. So, what will happen is this:

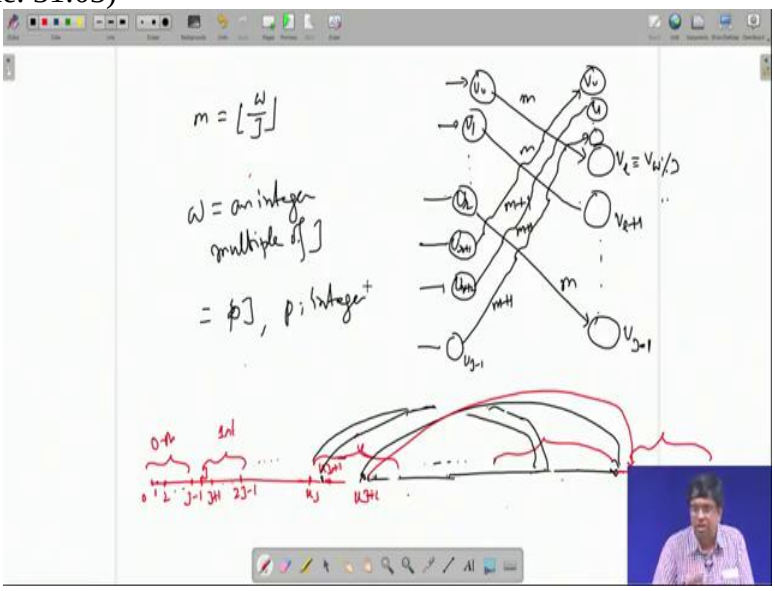
(Refer Slide Time: 29:47)



U0, let me erase this part first, U0 because I am handing KJ plus 0 to start with, that is taking me to some remainder here, so, that many index, that index V processor of that index, which is basically I plus w percentage J, I is 0 in this case, so, you have w percentage J, divide w by J, quotient number of delay and remainder is the index of V. So, that will come somewhere here, come V w percentage J because I is 0.

Then you take the next one, next one is KJ plus 1. If you jump from here by one, here also you go to the next. So this will go to next guy, I need to shift this diagram, I need space actually.

(Refer Slide Time: 31:03)



Here U0, V0 this is going to somebody, let me call it V l which is equivalent to V w percentage J. Then if you go to next one, KJ plus 1 is U1 that is processing and the output will be going to the same system cycles say under the same umbrella, so, number of system cycle delay will be the same. Here maybe system cycle delay suppose is m, m is nothing but w by J, I is 0 this, so m number of delay. KJ plus 1 data is coming out of U1. If it is shifted by 1 here, it will be ending up 1 later.

So, it will go to the next processor because the remainder will go up now, it will be V l plus 1, it will go, but still the number of delay will be same because from the same, from the system clock you are going to under the, going to the same system cycle, the same umbrella from where it colored. So, gap is same m and it will go on for a while till you reach a point for when you jump and you go to the last point, that is some index will come maybe U r that will take you to the last guy, last guy is VJ minus 1 that will handle this data.

Still m number of delay, then if you go further to the right because you have to cover the entire block, if you go further to the right, you will hit the starting of this. So index is 0, starting index but one block further to the right means delay will go from m to m plus 1, because we are going to the next system cycle now. So, next guy U r plus 1 will go to V 0 because starting index is 0, that is dedicated for V 0 so, it will go to V 0, delay is m plus 1.

Then you go further to the right and this also will get shifted to the right so U r plus 2 it will go to the next guy, V1 with the same m plus 1…Finally the last guy and last guy here UJ plus 1, K minus 1 sorry, it will go here m plus 1. So, there will be one set of edges with m number of delays and other set of edges with m plus 1 number of the system delays. You just think about what happens if W an integer multiple of J, that is W may be equal to say P into J where P is an integer, what will be the structure? You think about this, we will start from here in the next class. Thank you very much.