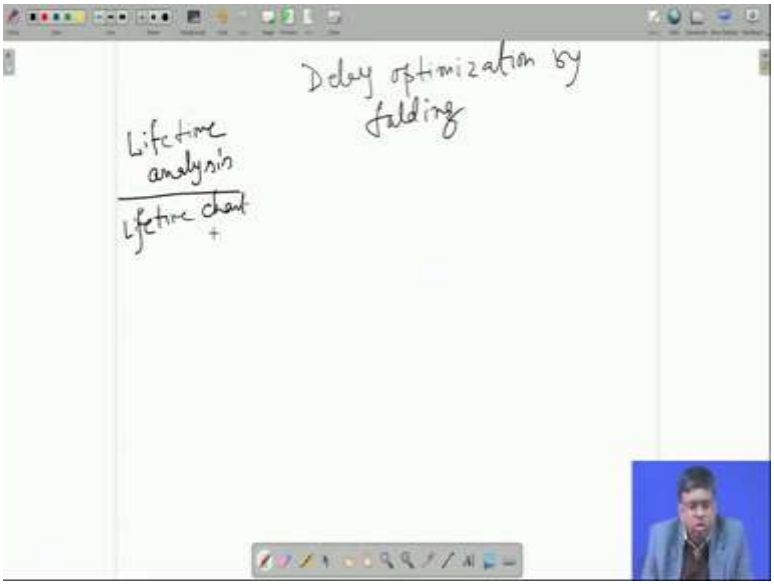
VLSI Signal Processing
Professor Mrityunjoy Chakraborty
Department of Electronics and Electrical Communication Engineering
Indian Institute of Technology, Kharagpur
Lecture 24
Introduction to Delay Optimization by Folding

Okay, so long we have did the folding where we have been tried to minimize the hardware that is the, there are so many adders, we tried to execute capital N of them by one hardware array you need. So while there are so many multipliers, so try to implement them okay n of capital N of them by one hardware multiplier you need so on and so forth. So it is folding by capital N, But the circuit otherwise uses lot of delays 1 delay, 2D, 3.

If you recall that folded circuit you will see I have got D, then 2D, then 3D, then 5D so there are so many delays. Next is how to optimize the number of delays also after having optimized the hardware units that is how to use, how to make sure that we are using minimum number of delays, so it is called delay minimization.

(Refer Slide Time: 01:10)

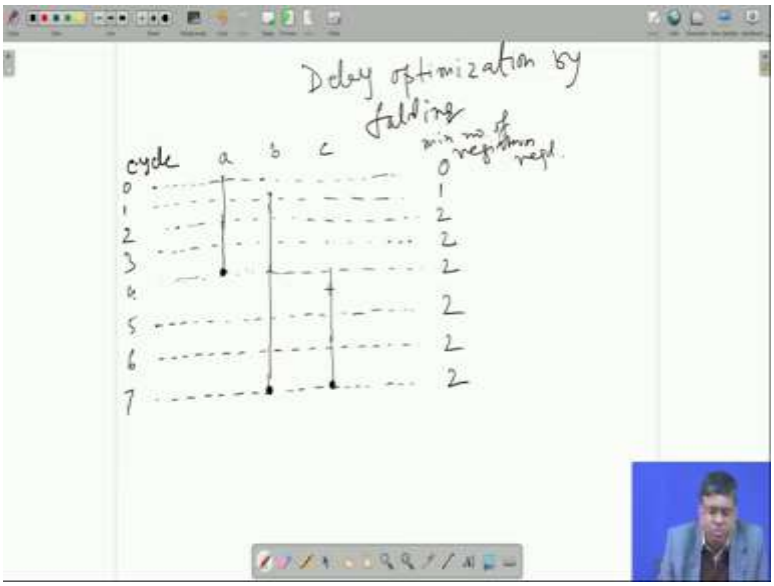


Delay optimization by folding. Okay, now what does a delay do? If it is one delay it stores one variable for one cycle then the variable is used up, it is lost. If it is 2 delay then that variable is stored for two cycles, during these two cycles it is intact. It is not change. Bur after the two

cycles it gets loose somewhere it is lost. If it is four delay then for four consecutive cycles, the variable is stored in the system without any change remains intact. After that only it gets used up.

Okay, so for delay optimization, we have to do some kind of you know some analysis where a variable have been examine, what are the variables that are present in the system, in the architecture? And for how many cycles each variable needs to be stored before it gets consumed? Okay, for that there is an analysis called lifetime analysis which begins with something called lifetime chart all right.

(Refer Slide Time: 02:50)



Now consider a system, the system suppose the system just generates 3 variables a, b, c in certain cycles, clock cycles, okay and we describe this generation so suppose system variable a is generated by the system at certain clock period. It is stored in the system by some delays up to certain clock period and then used up. B also generated the certain clock period stored in some delays and a particular cycle used up, same for C.

We describe this through a chart called lifetime chart. The structure of the chart is like this. You write cycle number here and then do like this, dotted lines, the cycle number 0, cycle number 1, dotted lines, this is the format, the chart. These are cycle numbers alright. Suppose there are 3 variables a, b, c I will write a here, b here, c here.

It is indicated so a is born that means a comes in the system, it comes alive in a particular cycle called 0. The cycle in which the variable is generated in that cycle no need to store it, it is just generated. I do not need to engage that delay but suppose it is giving like this a is generated here that it remains alive that is it remains stored in the system without any change remains intact in cycle 1, cycle 2, cycle 3 like that.

So I indicate by a vertical line. So in this example it goes up to 4. In cycle 4, it is still held in some flip flop. But then in during that stage only when it is held in a flip flop, it is used up somewhere, so from next cycle onwards, no need to store it. So then we indicate by a bubble. So I do not need any delay register to store a in cycle 0 but I need a delay register to hold it in cycle 1, then cycle 2, cycle 3 then in your cycle 4.

After cycle 4, I say the variable is dead, it is consumed. Similarly this example, b is generated in cycle 1, no need to store it in a flip flop in a delay then but it remains in the system in this example in cycle 2, cycle 3 onwards. With every cycle, I need to engage one delay to store b, when it goes like this.

In this example, it goes up to cycle 7 and gets consumed after this no need to have a delay to store b and there is of c, it starts in this example, it starts at 4 and goes up to 7. Here I write minimum number of registers required. Now listen, in cycle 0, I do not need any registers because b and c are not there at all. A is just generated but inside that cycle I do not need to engage a delay register to store a. So here it is 0.

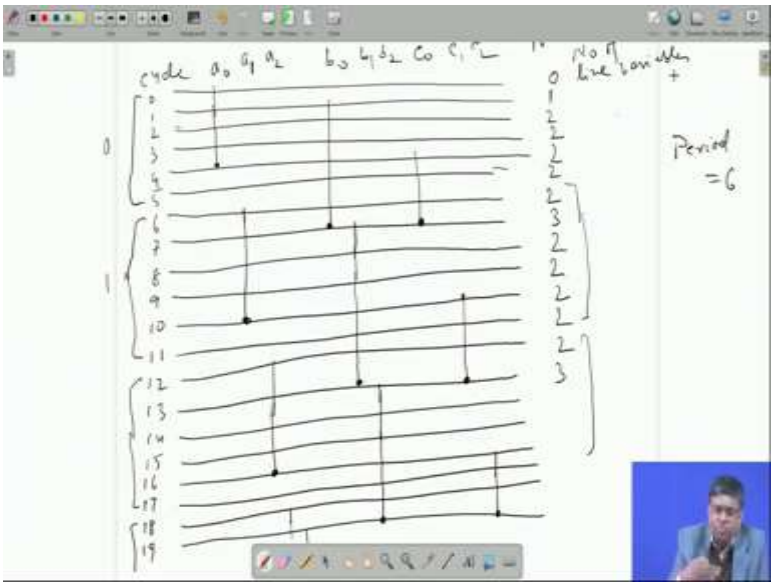
In cycle 1, a is to be stored so I need one register, b is just generated so for b I do not need any register. I do not need to engage any register to store. So I need only one. In cycle 2, I need one to store a, I need one to store b there is no c, so 2. In cycle 3, again one for a, one for b so I need 2. In cycle 4, I still need one for a, one for b, c is just generated so no need for c I do not need to engage a register to store c in this particular cycle.

So one plus one 2 again then once I go here, then again one and one 2, one and one 2, and one and one 2. So minimum number of registers required is 2. If I am giving two registers I can push data, I can manipulate data in them, but my storage will be over like the 2 registers out of the, what will be used to store a in these two cycles, then both will be engaged one for a, one for b up till then.

Then what will be relived of a, but will be engaged to store c, so two will be storing b and c like that, okay the minimum number of registers required is to found. These are very simple example, but it is to be found for our purpose but that many registers is unavoidable. So in my folded circuit when I am doing delay optimization there is a minimum number of delay registers I must have that I have to find out.

Here in this for this very simple example this one, it is but suppose like in a DSP, everything is to be did. There is one cycle, suppose one cycle a, b, c are generated like that. After several cycles, I mean one block of such cycles, I have got 1 a, 1 b, 1 c. After that block is over new A, new B, new C can come and so on and so forth. And there can be overlap between current block you know a, b, c and the new block a, b, c. Let me give you an example.

(Refer Slide Time: 09:21)



Suppose I have got cycle like that, this border right I am drawing this continuously, it does not matter. Okay this is minimum number of registers required, there is actually any cycle number of variables are live. Live means which are to be stored, which are not dead. Okay that is equal to the register number required.

Suppose I have got a period, period is 6, so 0 to 5, 1 period 6 to 11, 1 period, 0^{th} period I have got a0, b0, c0. This was a b c, now a0, b0 c0 they come as before. It starts at 0 and goes up to 4 as a previous example. B0 follows B of the previous example this generated here goes up to 7 and c0 so 0 starts at 4 goes up to 7.

So I would have been happy here, but then now you see this periodic, this is period 0, this is period 1. In period 1 the same phenomenon will occur, new A, new B, new C will get added so it will be a1, b1, c1 following the same pattern. So a1 will be generated in the cycle 0 of this, like a0 generated in the cycle 0 of this period.

A1 will be generated in the cycle 0 of this period which is cycle 6. So it will continue from here, it went up to 4 here so now it will go to 6 plus 4, 10. Thus 6 is added it was 0 earlier now it is at 6 that is a0 got generated in cycle 0. So you are at cycle 6, it went up to 4, so it will go to 4 plus 6, 10.

Similarly b1, b0 got generated at 1, so b1 will be generated 1 plus 6, 7 here. These are cycle 1 equivalently 7, line 7 is equivalent to line 1 there. Okay, so b1 will get generated, and meanwhile b0 went up to 7, so this will go up to 7 plus 6, 13. Similarly c0, c0 came at 4, so c1 will come at 4 of task 6 that is 10.

Yeah, because this 10, line 10 is equivalent to line 4, line 10 of period 1 is equivalent to line 4 of period 0. So c0 got generated at line 4, cycle 4, c1 will get generated at cycle 10 and it went up to 7, so now it will go up to 7 plus 6, 13. See already there is overlap, take one more variable a2, b2, c2.

A2 will be generated like a0 generated in cycle 0, this was generated at cycle 6 that is the starting cycle. So here also to be generated in the starting cycle a2. So it will go up to what earlier it went up to 10. Now it will go up to 10 plus 6, 16. Here b2, b1 got generated at cycle 7, so b2 will get generated at 7 plus 6, 13 alright, so here. And this went up to 13 now this will go up to 13 plus 6, 19.

Similarly c0, c1 got generated in 10 so c2 will get generated in 10 plus 6, 16 and go up to what? This went up to 13. So this will be up to 13 plus 6, 19. Again, if you start at 18, another one will start maybe a4, a3 start as we start like that, it will go on. So now you see because of overlap this minimum number of registers we got will change. Okay How?

In the first one there is no problem because there is no overlap from top where the starting one. So here as we put 0, here as before 1 for this, then 2 for this, there is no change, 2 for this here, one and one 2, 2 for this, then one and one 2, 2 for this, one-one 2, one for this because this is

just born in this cycle I do not need any register to store it, so still 2. But now see here one-one 2 I need one more here 3 and none for this because it is just born here so 3.

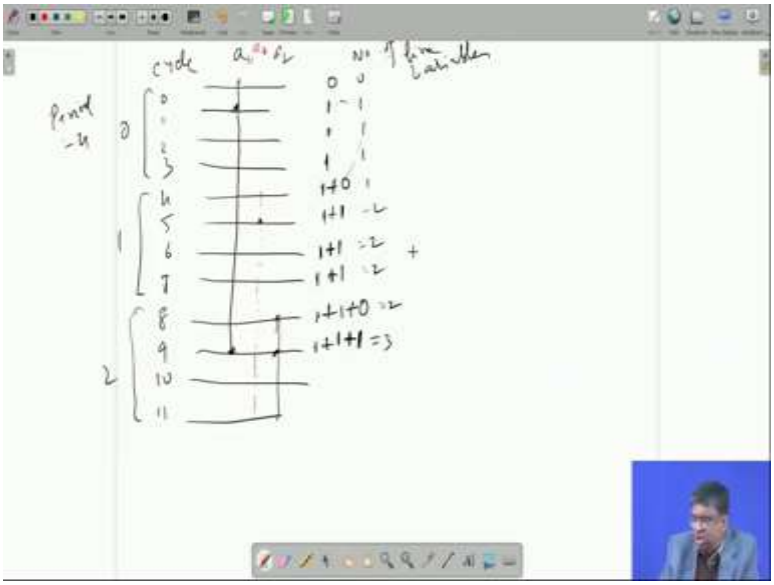
So this figure was too earlier. Now it has become 3 because of the overlap between period 1 and period 0. Okay next one again 1 plus 1, 2 this 1 plus 1, 2, this 1 plus 1 and none here 2. This is 1 plus 1, 2, this 1 plus not an array 1 plus 1, 2 and again 1 plus 1 plus 1 and 1 for this because it is just born so 3. And you will see this is repeat. This will just keep repeating.

After every 6 the same phenomena will repeat. First 6 will move 0, 1, 2, 3, 4, 5, 6 from here if you go they say that 3 will repeat 2, 3, again 2, 3. This will go on because of the periodicity. So now minimum number of register required will be actually 3. In fact it will be better to change this title instead of calling I mean now I have got multiple periods.

So I would rather prefer to change this to a better name more appropriate name that is number of I can call it number of live variables for which I need storage. So 3 is the figure maximum, which means I need 3 delay registers where number of live variables means number of variables to be stored.

So maximum is 3, so 3. Now point is would you really go and calculating like this, I mean period after period after period and calculating. There is a smarter way of doing it.

(Refer Slide Time: 18:08)



Let me tell you suppose there is only variable a, then number of cycle, number 0, 1, 2, 3 suppose period is 4 as an example period is 4, so this is one period and this is your a, then 4, 5, 6, 7 there maybe 8, 9, 10, 11 like that I will write number of live variables. Okay now what is given for one period only this is period 0, period 1, period 2.

For one period only the lifetime of a is given say. Say a goes for this period, as said it goes up to this. Okay, now here is a starting one. You write 0 then 1 you get 1, again 1 again 1, 1, 1, 1, 1, 1, 1, 1. Now a repetition of a you see will occur down maybe a0, a1 it is a0, a1 will come so it will come from cycle 4. It will go down like this.

So what is happening is this in this cycle I still have one, but here 1 plus 1. So what I have to do, this is 0, 1, 1, 1 first period I will leave as it is then when it is 1, then I look at this line one and go back by 4, step backward so 0. This 0 I will pick up, this 0 is standing for this guy which got generated here. It does not require anyone.

Then what is one? Again from here, this line I go back to this one. I have one so pick up one where from this one is coming? That is in this guy is generated this point which is equivalent to this point. Here we started here, then one point down, one cycle down we started here one cycle down 1 plus 1. Then if I move on go further up 1 plus 1.

If I have here further up 1 plus 1, 1 plus 1, 1 plus 1 we just have to go for along this line 4 up 1, 2, 3, 4. This one I pick up here, like in this case, 1, 2, 3, 4. This one I pick up here. Okay because they all correspond to this line. I am keeping them bringing the data for this line from this first one a0 only. I am picking the figure.

But then at cycle 8 again another person will come maybe a2 will come. So I will have overlap here again. So that means from 1 either go to, go back by 8, 4, 4, 8, this 0 corresponds to this new guy like this was starting here. This is starting here. So 0 for this, this is the 0 so I have got 0.

Then from here this one, I go back by 8 so I reach here, this is this point which is equivalent to this, this cycle that is after getting born next cycle, sorry after getting born next cycle here. Here it was 1 so for this also I take 1 so one more 1. Okay and now there is no further overlap. So now you get this figure 3, you get figure 2, 2, 2, 2. And this is 0, 1, 1, 1, 1 and from this only you can make out because afterwards whatever happens is a repetition of this because of the periodicity.

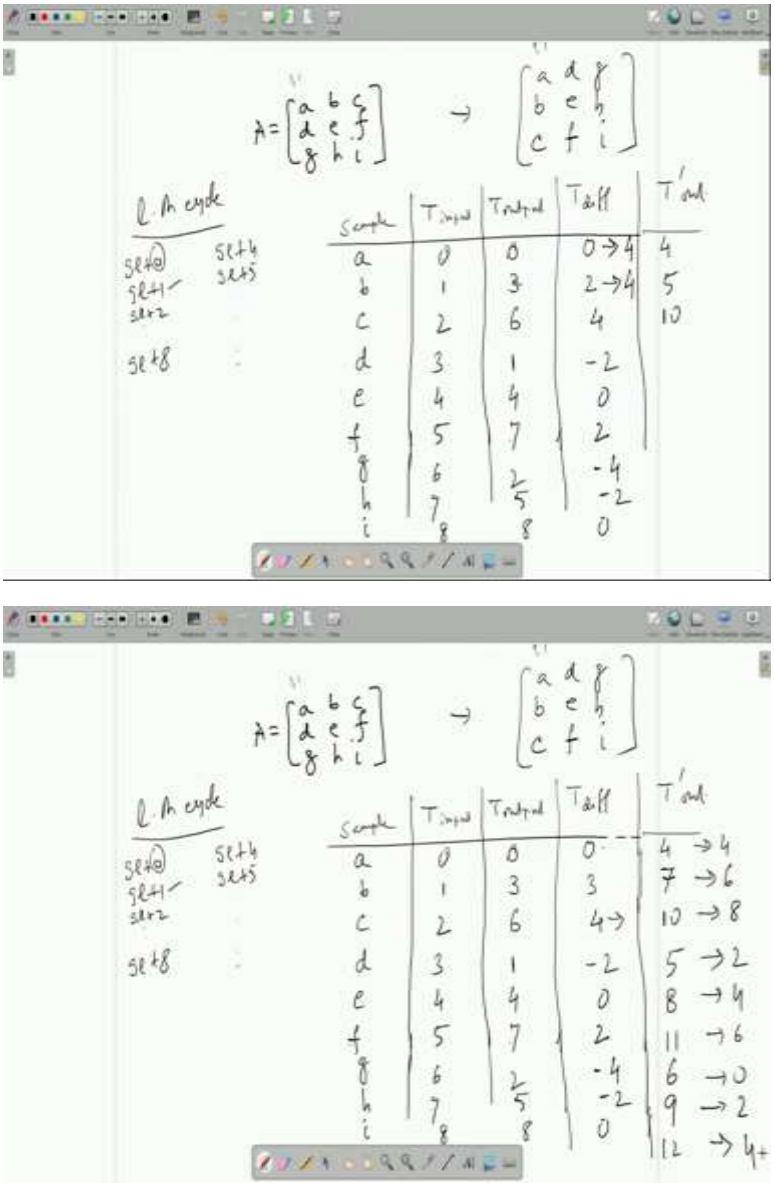
So in this case, I really do not have to draw the periodic repetition of a that is from a0 then a1 then a2 and going on period after period just from one period only going by backwards calculations backward backtracking and bringing back from appropriate places I can calculate this. This is smarter way of doing this calculation.

So minimum number of live variables is 3, which means I need that, I need minimum 3 number of delay registers to store them alright. But I here assumed that lifetime chart as though this a, b, c and there you know cycles of remaining alive and then getting dead and all they are given to me I was just calculating the number of live variables and from that if you got that magic figure that is minimum number of registers required to store them.

Even considering overlap between various periods that when the registers is good enough for the that figure I have got but question is who exhibits this lifetime chart once it is given, I can find out by that calculation that minimum number of registers required which the magic number, magic figured minimum number of registers required to store all of them. Even considering the overlap between various cycles, but who will give me this lifetime chart that you have to generate by analyzing the system.

I gave a beautiful example from Parhi's book, I do not know that you can complete it today.

(Refer Slide Time: 24:03)



But suppose you are giving a problem of matrix transposition a 3 by 3 matrix a, it is truly transposed to new matrix, what is happening is this at every lth cycle one matrix is given to the system, system is has a reader, the reader is moving at a faster rate, 9 times faster. And it has got lth original cycle will be broken into 9 small-small cycles 9 l plus 0, 9 l plus 1.

These are the system cycles like in folding after folding you have a faster clock, so here also system is working on a faster clock 9 l plus 2 up to 9 l plus 8, so 9 cycles. In 9 l plus 0 the reader

will come to place a, position a, it will read a. Then 9 l plus 1, it will move to b. It will read b, 9 l plus 2 it will go to c, so it is scanning like this a then b then c then d then e then f then g then h then i. These are your scanning and reading and there is a writer.

Writer will write this data in the matrix in appropriate places, by 9 l plus 0 it should read a. And that time, if reader and writer in the same position, they are synchronized, 9 l plus 0, reader also at 1 comma 1 position here, writer also at 1 comma 1 position. So reader reads a, writes a. Okay so what you have is this sample T input.

That is when you are reading, T output that is when you are writing as you can see they are synchronized, reader and writer moving hand in hand, when reader is at 1, 1 reads at 1, 1 when reader is at 1 comma 2, reading b this writer also at 1 comma 2 in this position for d is written and like that.

Okay if that b I will explain, first sample a, a will be written, a will be read at 9 l plus 0, so let me write 0. 9 l I am dropping if you want you can write 9 l plus 0. Okay and writer also in 9 l plus 0 standing here, but a is kept in the same place after transposition. So output will also be written in the same cycle at 0th cycle. So no need to store, you just read it immediately, right so differences is 0.

Then reader moves to b that is in cycle 1, 9 l plus 1 that time it moves to this position 1 comma 2, matrix 1 comma 1, 1 comma 2. So that time it reads b but b it will not write here. It will write b here, here when will it go? 9 l plus 0, 9 l plus 1, 9 l plus 2, 9 l plus 3 so it will be writing it at 9 l plus 3, cycle 3, so difference is 2 that means after reading b it has to hold it in some delay till the time comes for it, b has to be written the time is 9 l plus 3, that is 9 l plus 0, 1, 2, 3 that is what the writer does.

Writer is here at 9 l plus 3 that time only b will be used and b will become dead after that. So that means two cycles I have to store. Then come c, c means 1 comma 1, 1 comma 2, 1 comma 3. So that will be read at 9 l plus 2 but c will be written here so 9 l plus 0, 1, 2, 3, 4, 5, 6 so only in the 9 l plus 6 cycle writer will be in this place.

So it will be reading at 2, 9 l plus 2 by writing at 9 l plus 6, so 4 cycles it has to hold in the system this variable c after reading it. Then comes next guy d, 2 comma 1, in 2 comma 1 d so it

will be at 9 l plus 3, 9 l plus 0, 1, 2, 3 but d is to be written here. Why is the writer in this position? 9 l plus 0, 9 l plus 1 so 1, so now there is a problem. I love negativity, not causality.

I am reading at cycle 3 but I had to write at cycle 1 because d was to be written here. Unfortunately d I am writing only now 9l plus 3 but d was to be written at 9l plus 1 that is why I got minus 2. But I will take care of it.

Next is e, e is at cycle 4, 9l plus 4 here also 9l plus 4, no problem, 0 delay. Then f, f at 9l plus 5, here f 9l plus 5, 6, 7 so 7 so no problem I have to store it in 2 cycle. Then g, g has a big problem. G I am writing at, reading at 9l plus 6. But g is to be written here while writer is in this position that is at 9l plus 0, 9l plus 1, 9l plus 2, so cycle 2. So I am reading at cycle 6 but I have to write at cycle 2 so difference is minus which is non-casual again.

Then h, h I am reading at 9l plus 7 but h is to be written here 9l plus 0, 1, 2, 3, 4, 5 so 5 again negative. And lastly i, I is to be read at 9l plus 8, here also 9l plus 8, no delay, so how to get rid of this negativity? Suppose I create a timing difference between reader and writer what is the maximally negative here minus 4.

So suppose the writer starts at 4 cycle, after 4 cycle of the reader starting, reader started at 9l plus 0 then go to 1, 9l plus 2, 9l plus 3 but in 9l plus 4 that time writer starts suppose that means a will be read at cycle 0 but writer is starting its business at 9l plus 4. So writer is starting its business at not 9l plus 0, 4 onwards dot, dot, dot, dot. So at 9l plus 0 reader is reading a, it has to be in the system till 9l plus 4 because that is why the writer is coming into action, it will write.

It is pointing at 1 comma 1 at 9l plus 4 that time it will write a there, a was written read at 9l plus 0, so 9l plus 0 and now new value is 9l plus 4. So I now need the difference now becomes 4, for 4 cycle I need to store so I need delay. Then comes b, b was read at cycle 1 but here this will go to this position 1 comma 2, at 9l plus 5 because I started at 9l plus 4. So it will go at 9l plus 5, so 5.

Okay so earlier I had 3 minus 1, 2 now it is 5 minus 1, 4, so I need 4 delay. Then come c, c I will read at as before 2, 9l plus 2 but c earlier storing at 9l plus 6, now it will be 6 plus 4 because I am delaying everything by 4, so it will be 9l plus 10. Every cycle here will get added by 4, 0 plus 4,

3 plus 4. Okay, so let me restart again, a I am reading at cycle 0, at 9l plus 4 I am writing it fine. So difference is 4.

So from cycle 0 I have to store for cycle 4 then reader moves to this at 9l plus 1, writer originally had to store it at 3, okay the writer earlier was starting like this 9l plus 0, 9l plus 1, 9l plus 2, 9l plus 3 but now it is starting at 9l plus 4, so 4, 5, 6, 7 so it will be actually 7, so 3 will go to 7. That is why I am saying 3 plus 4 every cycle here it gets delayed by 4, 0 plus 4 is 4, 3 plus 4 is 7.

Similarly 6, okay earlier it was reading c, at cycle 2 but writing it when I am here that is 9l plus 0 1, 2, 3, 4, 5, 6. 6 will now become 10. 6 plus 4 because everything got started delayed by 4. So for earlier I reached at cycle 6, now I will be reaching at cycle 10. Okay earlier I had T output 3 now 3 becomes 7, okay 3 become 7.

Similarly earlier I had 4, earlier I had 6, 6 become 10, earlier I had 1 what is for 1, d. Okay I have to write here. So 1 now will become 1 plus 4, 5. Okay this is new T prime, a new T. So difference now will be 0, difference will be 2. Okay similarly this T out 4, we add 4 to it, it will become 8 everybody will get delayed 11, 2 plus 4, 6. 5 plus 4, 9 alright.

Now you see the difference, new difference will be what? 4 minus 0, so it will be 4, 7 minus 1 it will be 6, so I need 6 cycles. Okay similarly 10 minus 2, 8 cycles I need to store for 8 cycles. Okay cycle 2 I am reducing but I will write it at cycle 10, So 8 cycles. So 5 minus 3, 2, 8 minus 4, 4, 11 minus 5, 6, 6 minus 6 this is 0 earlier it was maximally negative.

Now I am brought down to 0, and this is 9 minus 7, 2 and there is one more i, earlier was 8 now it is 12, so 12 minus 8, 4. So now you see the difference is always positive so it is causal. Okay, I will start from here in the next class. Thank you very much.