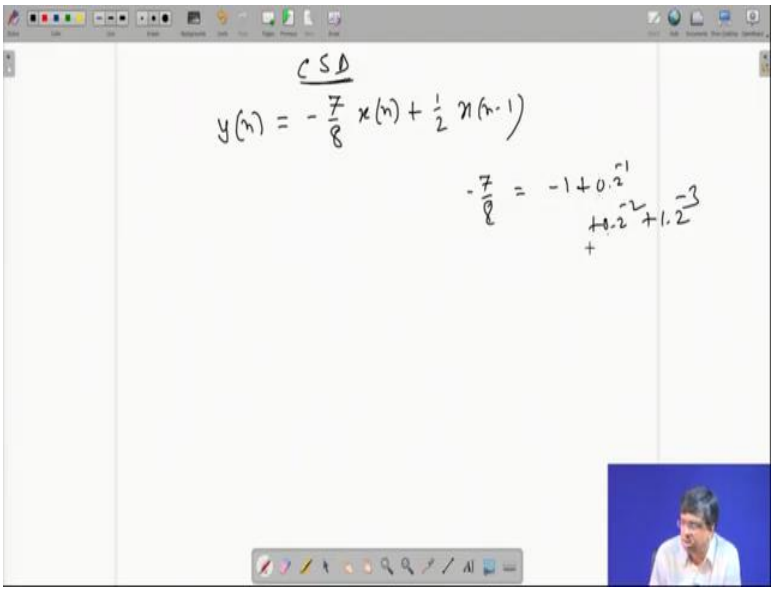
**VLSI Signal Processing**
**Professor Mrityunjoy Chakraborty**
**Department of Electronics and Electrical Communication Engineering**
**Indian Institute of Technology, Kharagpur**
**Lecture 37**
**Bit Serial Digital Filters**

(Refer Slide Time: 00:31)




Okay, so in the last class we discussed, towards the end of last class we discussed something called CSD, Canonic Signed Digit. Basically here in every position we can have either 1, 0 or minus 1. And, that is why it is called digit not bit because bit means only two possibilities, digits means more than two possibilities. But, this will give you the advantage.

There is a way of encoding a decimal number so that whenever you have got a non-zero digit like say 1 or minus 1, its neighbor to the left and its neighbor to the right will be 0. It can be done that way which means roughly a w-digit word will have almost 50 percent 0's.

And, these 0's help in reducing the multiplication complexity because in a multiplication table if you are multiplying some number by a 0 bit or 0 digit, it amounts to nothing, no change. So, you can skip that upper row. So, corresponding hardware also gets eliminated, that is a reason why it is done.

In fact, there are powerful tools today but for algorithms to convert a binary number, conventional binary two's complement number into CSD number and just you know, you can design even hardware for that, just gives up 16 bit binary input or n-bit whatever input and you get the corresponding CSD. I took some examples also and at that time I was taking purely integers like 11, 18 and things like that.

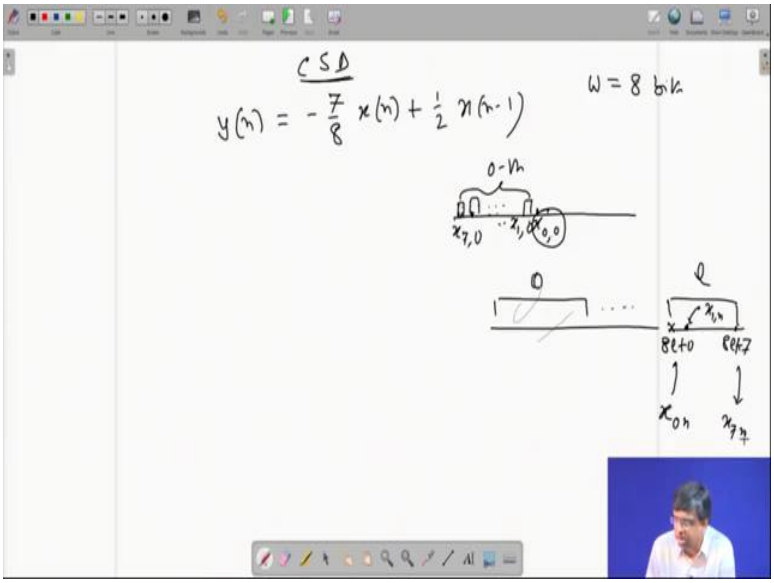
But as I told you we will follow, you know the convention when all the numbers are fractions between 1 to minus 1. So, suppose if a FIR has been filter has been designed, coefficients have been designed. They have chosen to approximate the coefficients by some numbers which are closest to that number but is in CSD form.

So, there is some approximation. One such example is, suppose output y n is minus 7 by 8 y I am sorry, x n plus half x n minus 1. Now, this coefficient minus 7 by 8 maybe the actual design had some other value close to minus 7 by 8. But that has been approximated to minus 7 by 8 because this is in CSD form.

Minus 7 by 8 is equals to minus 1 plus 1 by 8 that is 0 into 2 to the power minus 1, 0 into 2 to the power minus 2 and 1 into 2 to the power minus 3. So, minus 1 plus 2 to the power minus 3, minus 1 plus 1 by 8. That is how you get it, which means you see between these two powers, this is 2 to the power 0 and this is 2 to the power 3, I have got two 0's, this 0, this 0. So, this is in a CSD form though for fractional numbers. 1 by 2 is already a power of 2, so no question of converting it into further CSD, it is already in power of 2 form.

So, as I told you actual coefficient that we have designed maybe close to minus 7 by 8. So, that is approximated to its nearest CSD number which in this example is 7 by 8. Our task is to design a bit serial form of this FIR filter. We have already seen bit serial form of multipliers. We consider a multiplier called Lyons's bit serial multiplier. And, now we have to extend that philosophy not just to multiplier but more broader structured that is digital filters, FIR filter first and then IIR filter.

(Refer Slide Time: 04:24)




Suppose we are told word size, this important is important. It is 8-bit, means x n as well as y n that are 8-bit bus when the bits are coming parallelly. So, they are 8-bit lines that form a bus. But, we are now considering bit serial operation right, bit serial. That means there will be only one line through which the bits- LSB, next to LSB, next to LSB and dot dot dot up to MSB, they will come one after another for one word.

Okay, which means in terms of timing diagrams there will be only one line. You will have got one bit, next bit okay, dot dot dot. This is for 0-th word, 0-th word that means x0. You can say I have got LSB, x0, 0 okay, then x1, 0. This 0 stands for time, you can as well put it under bracket like this, bracket n dot dot dot x7, 0. This is 0-th word. So, 0-th word cycle is broken into, is divided into 8 bit cycles. This is 1, this is 2, this is 3 like that. So, in terms of index I put 1 index here, this okay.

Remember, I mean this is a first. The way we have drawn here, it is a reversal of time axis such though a leader. This is the leader; it comes first followed by this, followed by this, last is this. In terms of actual time, in terms of actual time I should have like this. This is the 0-th word this is the l-th word, okay.

In the l-th word first guy that comes at this location is 8l plus 0, that time which comes the LBS, LSB of l-th comes. Then 8l plus 1, next to LSB of x n comes dot dot dot 8l plus

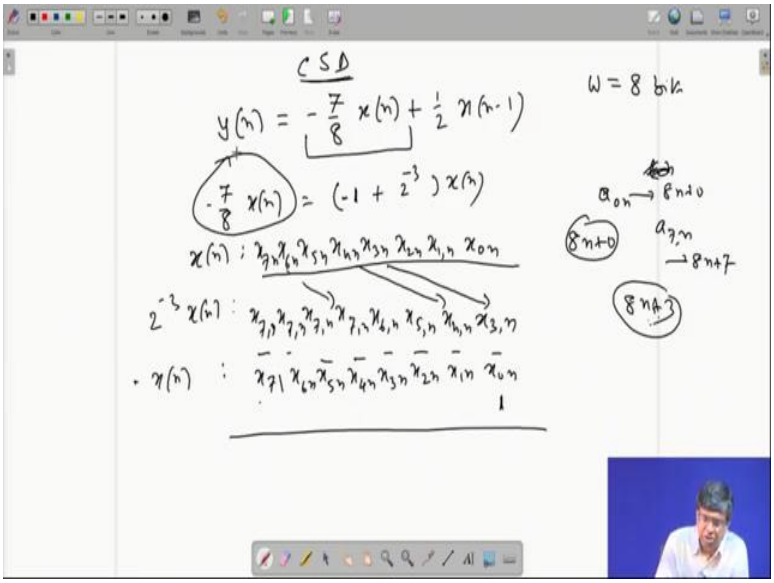
7. So, at this time who comes? LSB bits. I follow this notation as I have done here, 0 n alright 0 n. That is for the n-th data, xn n-th data; so n is here. 0-th bit, LSB and this is the x7n, MSB. So, x0n next at 8l plus 1 I have, at this point I have x1n dot dot dot dot.

So, that is how the bits are coming through one line. At 8l plus 0, x0n since the LSB of xn that gets in. At 8l plus 1 next bit comes in, x1n dot dot dot at 8l plus 7 last bit that is MSB comes x7n. This is all for the n-th word, so on and so forth.

But whatever we do, this job cannot be violated. This is the job. I must generate the same correct y n maybe by the bit serial method but should produce same correct y n. And, then in a bit serial system the output, the bits of y n also should come out through a single line, bit line serially; first LSB of that, y0 n, then y1 n, then y2 n, y3 n up to y7 n serially. So input will be x0 n, x1 n dot dot dot x7 n. In that order output also will be y0 n, y1 n, y2 n dot dot dot in that order. That will be a bit serial FIR filter.

But remember this basic equation cannot be violated, this is the actual digital filter. So, whatever we do inside bit serially, you know all those things, all those manipulation but this basic equation has to be satisfied. I should generate the same y n which the bit parallel that is word serial system ordinary FIR filter was generating, fine.

(Refer Slide Time: 08:31)



So, let us see minus this part first, minus 7 by 8 x n. Minus 7 by 8 x n as I told you, this is a CSD Canonic Signed Digit form. If you write like that, minus 1 into 2 to the power 0 plus 1 into 2 to the power minus 3, this is minus 7 by 8; this into x n. So, suppose x n, the 8-bit word is like this x0n which enters the system at 8n plus 0.

And, here I made a mistake. I said l and n; that l should be actually n. If I say a0n, a0n should come at 8n plus 0, a7n, I just made a mistake. Here I wrote n, here I wrote l there is no l. a0n will come at 8n plus 0, a7n will come at 8n plus 7, a1n will come at 8n plus 1, a2n will come at 8n plus 2.

There is no l. Last time I wrote l here and n here, I am sorry for that mistake. Both should be n because it is the n-th word. All the bits of n-th that are now bit serialized and the cycles are 8n plus 0, 8n plus 1 up to 8n plus 7. And 8n plus 0, an0 goes in. At 8n plus 1, a1n goes in so on and so forth.

So, this is x0n, then next bit is x1n, then x2n no comma is required, x3n, x4n, x5n, x6n, x7n. These n bits, they will come into the system serially. This will come first at 8n plus 0 here at this cycle. Then, this will come at 8n plus 1, this will come at 8n plus 2 dot dot dot, this will come at 8n plus 7 fine.

But what I have in the equation? 2 to the power minus 3 into x n to be added with minus x n. So, 2 to the power minus 3 x n bits, the bits will be shifted to the right by 3 bits and also that sign extended to the left by 3 bits because it is 2 to the power minus 3. Remember, whenever you define a number by 2 that there is 2 to the power minus 1, if you multiply it by 2 to the power minus 1, all the bits get shifted to the right by 1 and the sign bit comes back. This is a sign bit, x7n, MSB that again comes back to the sign position, MSB position. That is called sign extension.

So, if you do it thrice, this will occur thrice. So, what will happen is, so this will go back out, this will go out, this will x3n this will come to this place, x3n. Okay, that will be the new word. Then it will be here, x4n. x0n goes out in first shift only, x1n goes out in the second shift only, x2n goes out in the third shift only. And, in the third shift x3n comes to this position, LSB position, then x4n, then at this location x5n, then here x6n, x7n.

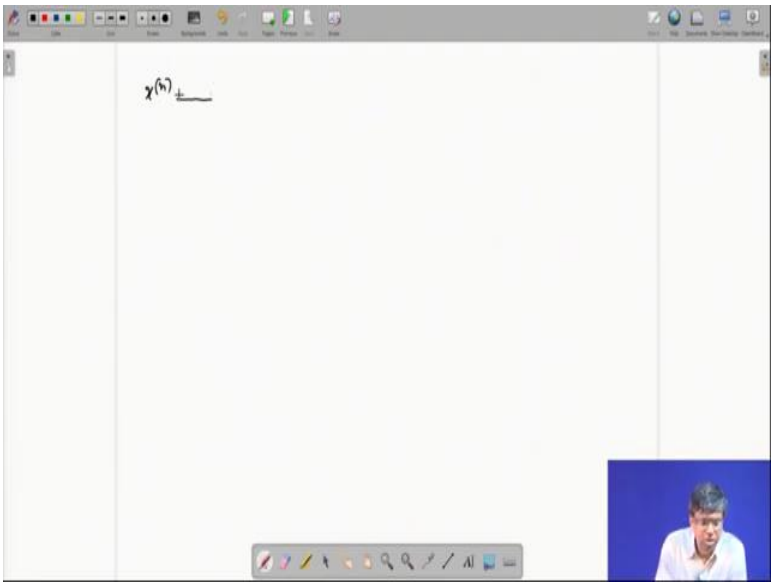
After that the sign bit will get extended here, x7n one extension, second extension, this is third extension, alright. This is 2 to the power minus 3 x n. I have to add minus x n with that. Minus x n means I will take the two's complement, means there is minus x n means the original word have to take the two's complement. So, x0n should be complemented, x1n should be complemented alright and 1 has to be added at the LSB.

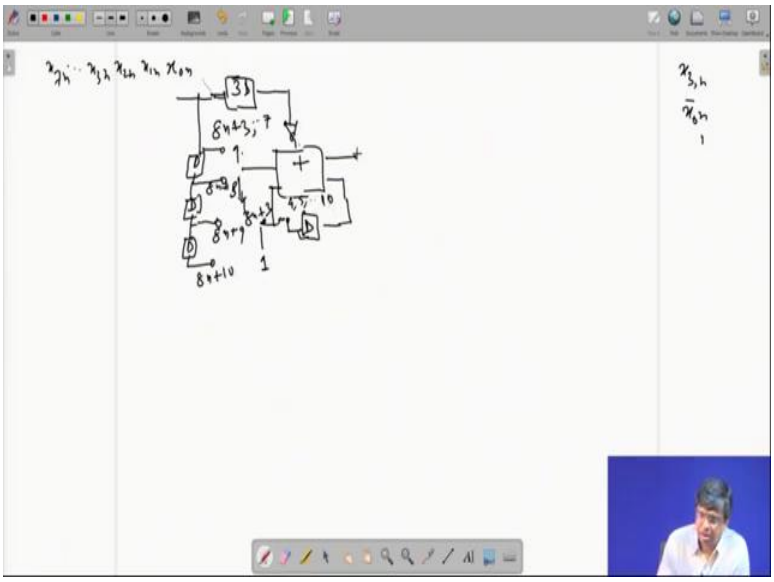
This is the job I required to do to just get this value minus 7 by 8 x n 0 and we will do that serially that is I will have a system where x0n will come in at n8 plus 0, x1n will come in at 8n plus 1 and so and so. And, this result will come out bit serially. Now, you see, first who am I adding? I am adding x3n with x0 bar n. Remember, x3n comes to the system at cycle 8n plus 3. At 8n plus 0 x0n comes, at 8n plus 1 x1n comes, at 8n plus 2 x2n comes, at 8n plus 3 x3n comes. So, I have to wait till that time. So, at 8n plus 3 now I got this.

And, then x0n which came 3 cycles ago, at 8n plus 0 that I have to use now which means that x0n I have to store in some flip flops. How many? Three, because one cycle, second cycle, third cycle for three cycles I have to hold so then as a compliment, so at 8 plus 3 while this x3n comes and got these three available.

You add them and with taking 1 as initial carry in the adder. You get a result that is let me come at 8n plus 3 new carry will come to this side. Again add, again the result and so on and so forth. This is what you will do for this part.

(Refer Slide Time: 14:38)





So, let us see how the circuit will be okay. At 8n plus 0 LSB came, then at 8n plus 1 next guy came, 8n plus 2. So, you can with show this here. This all way things are coming; x0n, x1n, x2n, x3n, dot dot dot x7n. It came at 8n plus 0, then 8n plus 1 so and so. And as I told you, I have to add x3n. This is the thing I have to do with x0 bar n with a 1. X3n as I told you last time it comes at 8n plus 3.

So, that is the time my job will stop start my work will start. I need x0n then bar. So, that means x0n which comes here, I have to store it. I have to make it wait through, 3D and

then there is a NOT gate. I have to add, so this comes here. Let me have a switch. This switch will start at 8n plus 3, at 8n plus 3 it will touch here. So, I will get x3n and that time I will have x0n bar coming here and this adder, its carry. Initial carry is binary 1 and that is at 8n plus 3 of course. That is the time we are adding start. New result comes, new carry, okay.

Then, at 8n plus 4 next guy comes, x4n and that time if you see this comes at x8n plus 4 that time I need the next guy x1n. So, x1n came at 8n plus 1 so after 3 cycle delay it will be available when x4n has come. Similarly, x2n which had come at 8n plus 2 after 3 cycle delay; okay it will be available that time I have x5n coming because that will be cycle number 5 so on and so forth and all will be complemented.

And this will go on till what time? Till I have no extension coming, up to here, x7n. x7n comes at 8n plus 7 here, and this fellow came, x4n came 4 cycle ago after 3 delays, 3 flip flops. It will come at the same time, cycle number 7 and complemented you add. Okay, up till this there were no problems.

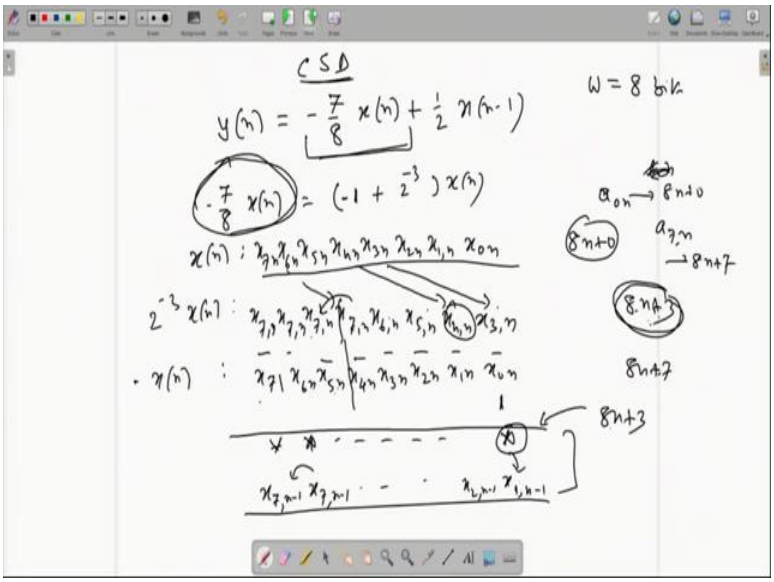
So, 8n plus 3, 8n plus 4, 5, 6, 7. These are the 5 cycles in which I go up to this, no sign extension required. So, this will be 3 to dot dot dot 7. Then what happens? And always as you go on, previous carry will be held in flip flop. It will come here. Okay, 8n plus 3. Here it will be from 4, 5 dot dot dot and I will complete it later.

Okay, so up to 8n plus 7 no problem. After that what happens? 8n plus 7, the same guy is extended. At that time x5n; okay this is cycle number 8. This is cycle number 7, cycle number 8. So, 3 cycles ago x5n came. After 3 delays, 3 flip flops 3 delays, in the cycle number 8 this will come up, complemented and that time this will be coming back as the extended, sign extended and I will be adding them.

Okay, then again further sign extended the same thing will be continued. So, next bit so that will be cycle number 9. So, 6 cycles ago x6n came. So, after 3 flip flops, it will come in the same cycle 9, complemented, added. And again this will come at cycle 10. Same thing will get extended further. 3 cycles ago x7n came. Now, it will be surfacing after the 3 flip flops, added right.

So, if you do that 8n plus 8 this is how it will go we will go. It will touch it here. This carry thing will be like this, previous carry moves to new carry; moves to next (())(20:39). Again once you add something, hold the carry in the flip flop, pass it to the next cycle. So 4, 5 it will come up to 10. So, this is where this quantity will come up, bits of this will come up, bits of this.

(Refer Slide Time: 20:53)



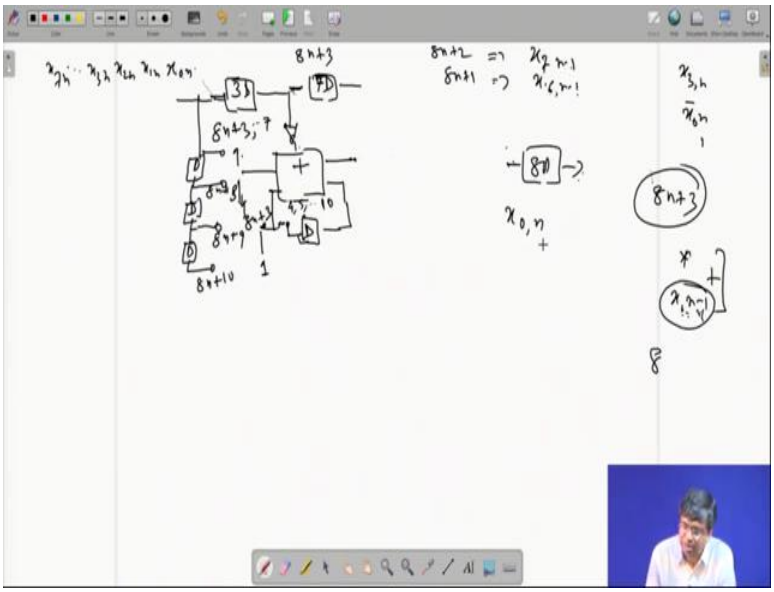

We started at 8n plus 3. This is the cycle; we started at 8n plus 3 alright. So, this will come this fellow, this will come at 8n plus 3 then 8n plus 4 like that through that adder output. With that now I have to add half x n minus 1, half x n minus 1. So, half x n means, for x n I wrote the bit like this x0n, 1n, 2n like this.

For x n minus1 it will be the same, x0 n minus 1, x1 n minus 1, x2 n minus 1 dot dot dot. This could be 1 or 0, this could be 1 or 0, this could be 1 or 0. But they are not same for both the numbers, x n and x n minus1. And then that you are multiplying by half. So, everything will be shifted to the right by 1 and there will be one sign extension.

So, it will be instead of, instead of x0 n minus 1, after it shift to right by 1 because of this half multiplication that will go out. X1, n minus 1 will come here. Then x2, n minus 1 will come dot dot dot x7, n minus 1 okay. And, then this will be extended and then you finally get the result. So, this part, alright, this part we have to do again.

Now, this is coming out at 8n plus 3. With that I have to align this number that is I have to make sure that at 8n plus 3 this is available so that I can happily add them with initial carry 0. Nothing is there, how to do that? What is this? X1, n minus 1. Let me go to the next page.

(Refer Slide Time: 23:26)



So, our purpose is this, at 8n plus 3 whatever is the output, adder output that is denoted by star and I need to add this these two to be added. So, I need to do this alright. That is for n minus 1-th word, number 1 bit from where from I get, that is the question. Now, you see this is a very tricky thing.

At what time the LSB of this come? 8 n plus 3, x0n comes here at 8n plus 3 and at that time I get the result here, okay? At 8n plus 2, what was available here? It was the MSB of the previous word, n minus 1-th word, is not it? They are coming serially. X0n came at 8n plus 3, then x1n, then x2n and at 8n plus 4 x1n, at 8n plus 5, x2n, at 8n plus 3, x3n like that.

But if I go before this, at 8n plus 2. So, it no longer will be the n-th word. It will be a bit of a previous word, last bit MSB. So, that time what will come here at this point? X7, n minus 1 MSB, last bit. At 8n plus 1, who will come? x6, n minus 1 dot dot dot, alright?

So, I am standing at 8n plus 3. That time I have to add this with this guy, but x1, n minus 1.

I am getting x7, n minus 1 at 8 plus 2 that is if I delayed it by, just delayed it by 1 cycle then at 8n plus three, I have got this result here. I have got x0n here. So, here I will have the data which came here a cycle at 8n plus 2 which is x7, n minus 1. So, I will have x7, n minus 1. I will have this output.

But I am not interested in x7, n minus 1. I am interested in x1, n minus 1. If I increase the delay by 2, what will happen? At 8n plus 3, I have x0n. And, so what will come here at 8n plus 3? That only which was here 2 cycles ago, if I write it as 2 delay. Two cycles ago will be x6, n minus 1. One cycle ago was x7, n minus that is the last bit of the previous word. Two cycles ago last but one bit of the previous word, so and so.

But my target is neither x7, n minus 1 nor x6, n minus 1. It is x1, n minus 1. So, we have to go on delaying. How many cycles? We will see if I delay by 7 cycles then we will get this x1, n minus 1 here. You can also verify another way. If I delay it by not 7, 8 cycles then a very interesting thing will happen.

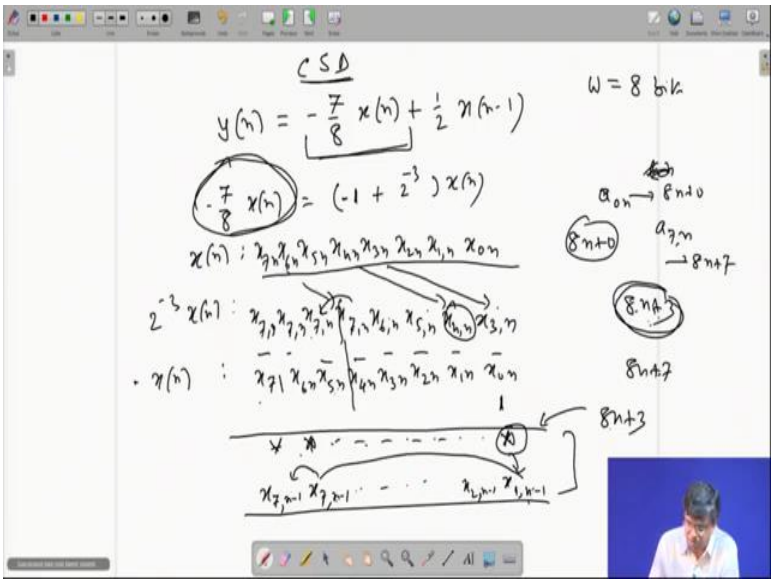
What will come here at 8n plus 3? That thing, which was here 8 cycles ago. Now, currently I have got x0n and data size is 8-bit. So, 8 cycles ago x0, n minus 1 would come. All the 8 bits of x0n, x1n you know there we will come later. If, I delay it by, this by 8D not 7D.

Suppose I delay it by 8D and this is a cycle I am standing. What is it I have here? I will have only that here which came here 8 cycles ago. Now, which came here 8 cycles ago? Now, at 8n plus3 what is currently available? Currently available is x0n. This is currently available.

So, 8 cycles ago who was available? It is periodic over 8 right? x0n 1 cycle ago, x7 n minus 1 because x0n is staring bit of n-th word. If we go 1 cycle ago, I am in the previous word, n minus 1-th word and the last bit. If I go 2 cycles ago, I am at n minus 1-th word, 6-th bit, okay. That way if I go 8 cycles ago, I will in n minus 1-th word 0-th bit. But 0-th

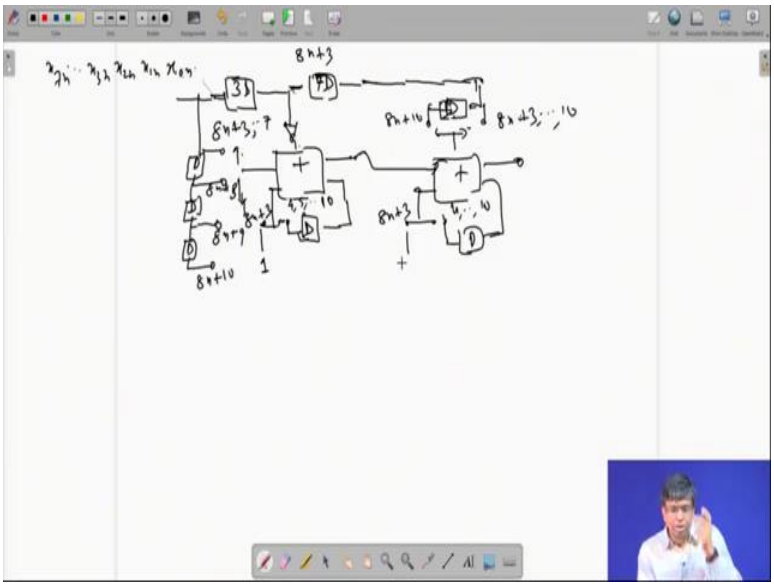
I do not want. I want 1. So, I do not need 8 delays, 7 is enough for me. Therefore, it is 7D. And rest we will continue. Okay rest we continue. So, I have them alright.

(Refer Slide Time: 28:33)




Now, look at this previous thing, this data and this. So, this will go on, up to this there is no problem. Okay, adder output I will get these bits, followed by this, followed by this followed by this dot dot dot and I will get these. And after the 7 cycle delay, the flip flop I have added I will get first these, then these, then these like this. Only problem will come when I need the extension, this field has to be extended. This was 8n plus 3, 4, 5, 6. What is this cycle? See 3, 4, 5, 6, 7, 8, 9, 10. So, at 9 whatever I get that has to be extended to 10.

(Refer Slide Time: 29:14)

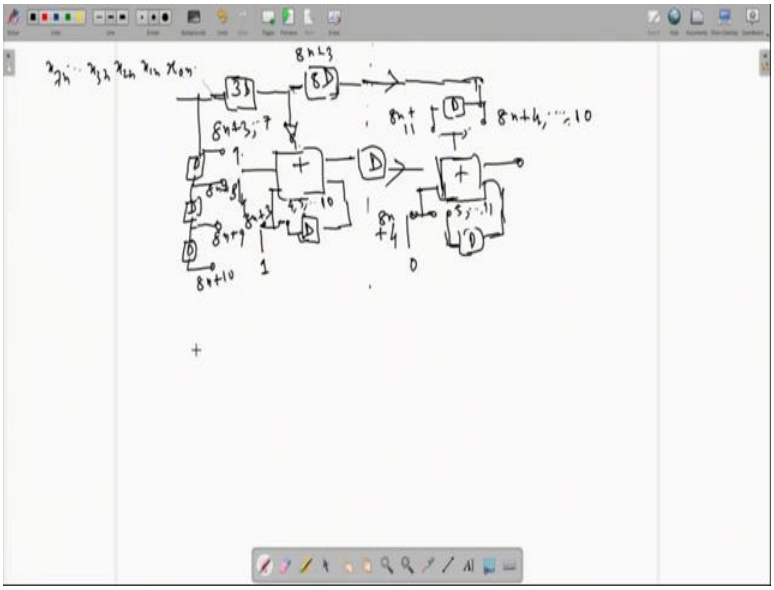


So, this will continue. There is a delay, okay. So, 8n plus 3, 4, 5, 6 up to 9, 8n plus 3 dot dot dot up to cycle 9 it will be here. And, it will be after delay 8n plus 10. Let me draw nicely and the carry, initial carry here is 0. So, I start the addition at 8n plus 3. So, 8n plus 3, it will be here result is here and always previous carry will be feedback here. So, 8n plus 4 dot to 10. So, this is a bit serial realization of FIR filter, okay.

Now, you can make some you know engineering manipulation over this. If you see, if you forget the time taken by the inverters, the critical path will be basically this adder which is in series which another adder, so 2 adder delay. But these are all in forward directions, forward directions. So, you can have a, you can take them as ages in the forward direction. You could apply delays.

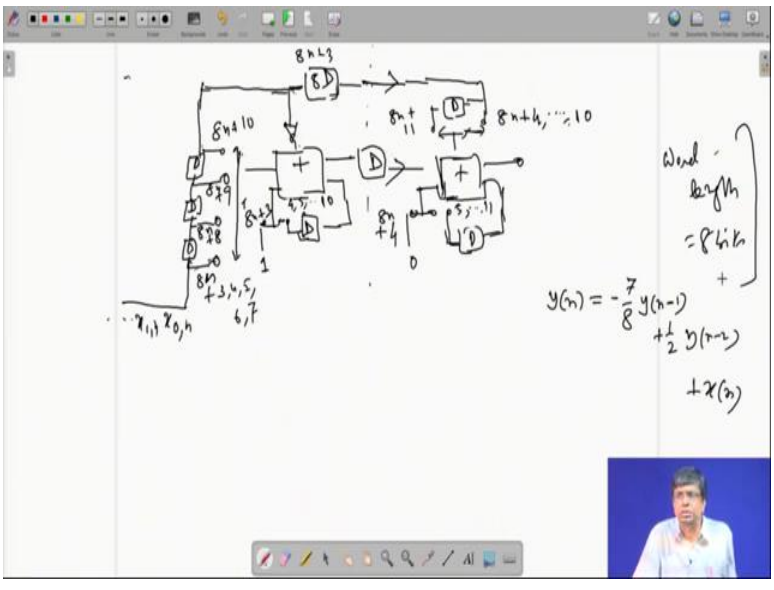
If you bring in some delay here, instead of 7D it will become 8D. You need a delay here, D. The moment you delay here, everything timing will be advanced by 1 mode, 3 will become 4 because we are delaying by 1. So, everything that has to be taken care is the timing here. So, 3 will become 4, 10 will become 11. This is 3 to 9, sorry. We can count again, 8n plus 3, 4, 5, 6, 7, 8, 9, yeah up to 9. So, this is 9.

(Refer Slide Time: 32:37)



But this timing will now change because I have added delay. So, earlier it was 3 to 9. Now, it will be 8n plus 4 dot dot dot 9, sorry 4 dot dot dot 10, and this will be 8n plus 11, alright. This will be 8n plus 11. We will draw it in a neater way. I am just trying to make it neater that is all. So, this is 8n plus earlier it was 8n plus 3. Now, it is 8n plus 4 and 5 to 11 will be here, 8n plus 5 to 11. And, you get the output, okay.

(Refer Slide Time: 34:50)



There is one more manipulation we can do. You can see the bits are coming here and they are getting delayed by 3 flip flops, 3 delays and the same bits are also subjected to 3 delays D, D, D-3D. So, why waste the 3 delays twice? Why not use the 3 delays only once. Instead of giving the input here, give the input through this path and node 3D here.

Now, before, you see here, before these bits come to the delay here, they only have a switch from cycle 3 to 7. So, if I give the bits here. Here this switch will be from 3 to 7. Then after 1 cycle, it was cycle 8; 8n plus 8. So, after 1 cycle this will be 8n plus 8. After 2 cycles, it was 8n plus 9, so after 2 cycles it will be 9 and this 10. So, it will be 8n plus 10.

So if you do like that simple engineering manipulations, so bits of x n are coming here- x0 n, x1 n dot dot dot. And, here it is 8n plus, as I told you 3, 4, 5, 6, 7. This is one switch. The next one, you can just shift 4, 5, 10 here, this 8n plus 3, 4, 5, 6, 7, this 8n plus 8, 8n plus 9, 8n plus 10. This is the situation, okay.

So, this x0, n comes here at 8n plus 0, then x1, n comes in 8n plus 1 and so on and so. But I start getting output at 8n plus 4. It was 8n plus 3 when I use (())(37:18) timing; it was 8n plus 4. It is bit serial realization. These are for FIR filter. How about IIR filter? IIR filter has a loop. And I as I told you, whenever there is a loop, you know, any operation and especially retiming and other operations are definitely more difficult because of the feedback. You know, many things have to be put in place.

So, you consider one IIR equation now. See, y n depends now on y n itself that is why it is IIR, y n minus 1 plus half y n minus 2 and x n. Again, we have 8-bit realization. Okay, we will assume word length, okay. I will go for a realization and that time we will discuss also that always it is not; I mean realization may be possible, may not be possible depending upon the word length and also the coefficients and other things. Okay, that we will so in the next session. Thank you very much