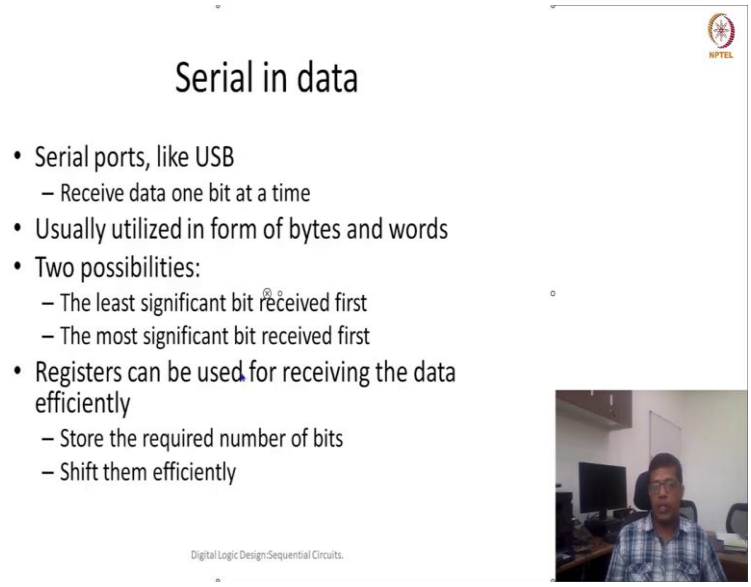
Digital System Design
Professor Neeraj Goel
Department of Computer Science Engineering
Indian Institute of Technology Ropar
Lecture 48
Registers-2

(Refer Slide Time 00:16)



So, let us go into some different applications, one more applications of these registers. Now, we have seen USB ports. So, these USB ports are serial ports. What do we mean by serial ports? You see, when you connect your USB drive, so, USB drive will have a large amount of files. It is, these days it will not come less than 4 GBs or 8 GBs or sometimes it could be even more than that.

Now, if you have seen inside or you have seen the port of this USB, you will see there are only four wires. Two wires are VDD and ground, that is why if you plug in your USB, it, your phone acts, this USB cable acts like a charger. The other two pins are there to transfer one bit of data at one time. So, because although your USB disk or USB drive may contain GBs of data but only one bit is transferred at one time.

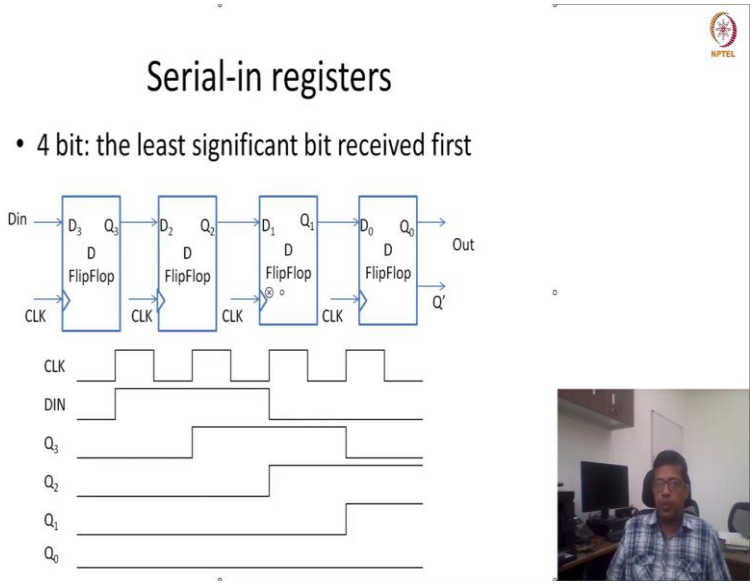
So, at the receiving end, we would like to have movies stored in our USB drive or maybe lectures stored in USB drives or maybe PPT stored in USB drive but it is being transferred bit by bit. Eventually, we would like to read the bytes, we would like to read the characters, maybe words, maybe whole files but it is one by one.

So, what do we do? How do you we receive this serial data? That is why it is called serial. Because data is being received bit by bit. Now, when we are receiving it bit by bit, there are two possibilities. Now, either we take least significant bit first and then most significant bit and then we can store them in registers and when enough number of bits are there, let us say, we would like to store 32 bit of data, so, once 32 bits has been received, then we can read it out as a parallel register or as a 32 bit number.

So, how can we do that? So, what has been seen, you can use certainly, flip-flops. But if flip-flops are connected in a certain manner or your, eventually you would like to store them in registers but these registers has to be connected in certain manner so, that we can read them efficiently. They are being fed one by one but we can read them efficiently. We can read all of the data in parallel. So, we can read, like, 32 bit of data at once. But they would be fed bit by bit.

So, can we use registers efficiently? Because the way we have created previous registers, they were parallel in, parallel out. You have to give, there were n numbers of inputs to my registers. D in was a n bit number. Here, can we have, the question here is, can we have single bit as an input here? So, let us see, if I would like to do it in registers, how can we do that?

(Refer Slide Time 03:53)



So, instead of taking 32 bit number which can be, like, whatever solutions that we are developing here, it can be scalable to that also, but let us take a 4 bit as a number. So, we are trying to store 4 bits in a serial in manner, one by one. Because there are two mechanisms,

either we can take first least significant bit or most significant bit. Let us take least significant bit first.

So, what we can do is, we can take this least significant bit and give it to a register, give it to a flip-flop. Now, in this flip-flop, we are giving this input, least significant input. I said, it is a 4 bit register so, I have 4 flip-flops. D3 is the most significant flip-flop. And I am connecting my D input, the single bit input, serial input to my most significant flip-flop.

The output of this flip-flop is connected to the next flip-flop, D2. Then the output Q2 is connected to D1 flip-flop, flip-flop at D1 position and the Q1 is connected to the next, the least significant D0 input of the flip-flop. So, there are four registers but they are connected in this manner that my output, so, my serial input is connected to the most significant flip-flop and then each output is going to the lesser significant flip-flop till the end.

Now, because output is connected to the next flip-flop, this is also, called Shift registers. And these are also, called Serial-in registers because there is only single input. D in is a 1 bit input and which is being fed to a 4 bit register. It is quite generic. If you want to take 16 bit register or 32 bit register or 64 bit register, it can keep on cascading like this.

Will it work? How will it work? So, all these questions which are coming to my mind. So, let us see by taking an example and by flipping the bits. So, let us consider a clock period and let us also, consider my D in clock or D in as an input. Now, further consider that my clock is positive edge is the active edge and if I see what is my input? My input is, this is 0, this is 1, 1 and 0.

Now, because this is the least significant bit. So, this is bit 0, bit 1, bit 2 and bit 3. So, if that is the case, now, let us see, what would be the, so, if this is the input, what would happen at Q3? So, in Q3, this 0 would be sampled at this time. Because 0 is sampled, my output for this 1 clock period is going to be 0. Now, in this active edge, one is sampled, so, 1 would be the output for next one clock cycle.

Next active edge, 1 is being sampled so, 1 would be the output here for next clock cycle. And at the end of this fourth clock cycle, because 0 is sampled this would be the output. So, this would be the output for my Q3. Now, Q3 is given as an input to D2. So, what would be the output at Q2?

So, at Q2, everything is shifted by one clock cycle. So, whatever was the input here, that would be the output after one clock cycle. So, you see, you can quickly check, so, at this positive edge, because Q3 was 0, Q2 would be 0 for one clock cycle. Similarly, at this positive edge, Q3 was 0 so, my Q2 output will be there, 0 for one clock cycle.

Similarly, at this positive edge because this was 1, output would remain 1 for one clock cycle and similarly at the, this particular positive edge, my output was, my input was 1, so, output would remain 1 for one clock cycle. Similarly, for Q3, Q1, my Q2 output is given as an input to this flip-flop as D1, so, D1 is Q2. What the output would be? Output is again shifted by one clock cycle.

So, 0 for one, first clock cycle, the second clock cycle, third clock cycle and here it is 1 because in this active edge, this 1 is being sampled. Similarly, if I see Q0, for Q0, for all the clock cycles it is going to be 0. So, if I see this way, I can see that this particular 0 has reached, at cycle number 4 in Q0. As a Q0 output and this is 1, this is 1, so, this is my bit 0, this is bit 1, this is bit 2 and this is bit 3.

So, we have received them in the reversed order. Although D in was connected to most significant register, most significant bit of my flip-flop but because it is cascaded, my output was connected to the next input, similarly the next output was connected to the further next input. So, that is why we were shifting the value of D in to the right direction, every time.

We can also, call this particular register as Shirt right register. We are shifting the output in the right direction every one clock cycle. Incidentally here, the input was only D in, only single bit was the input. So, we can, we could have designed like this. Now, one more question which comes to my mind.

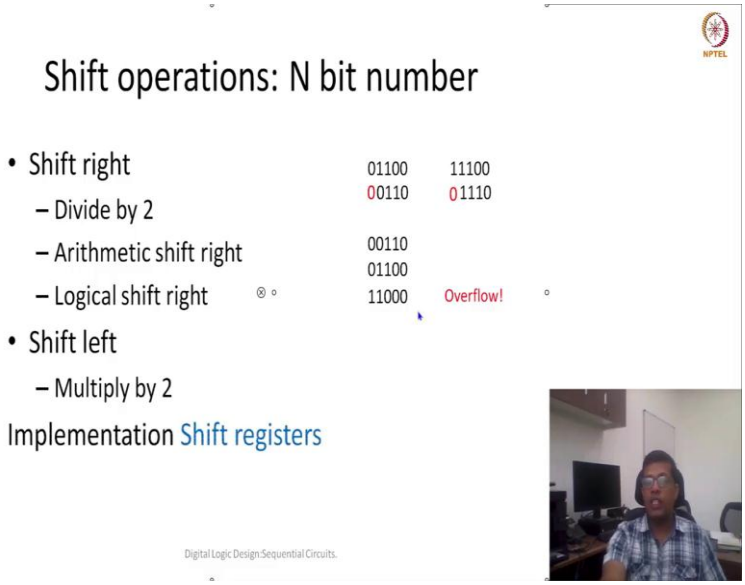
So, here, my clock as well as D in, both are shifting at the same time. How is it possible? And my output is still correct. So, here, my assumption is that my, my assumption is that my D, my hold time is 0m but setup time is not 0 and propagation time is also, considered as 0 to make this circuit look simple.

However, usually my, as D in would be, so, there would be certain delays involved here. So, because of those delays, my, I can make sure that, the setup time is non zero and propagation time is also, non zero, then also, it would work correctly. So, but to make this diagram

simplistic, we have taken setup time as well as propagation, sorry, hold time as well as propagation time as 0. Good.

So, this way, we can use this serial input as serial input but still we can use these registers. Now, Q0, Q1, Q2, Q3, from there we can always read out the values. We can always read out the values and we can read them parallely. So, whenever we would require to read this 4 bit value. So, let us say, it is a 32 bit register, then it would require 32 cycles because each data would require one clock cycle. After 32 cycles, we can read all the 32 bit data at once. And then reuse them wherever it is required. So, with this, let us move on to the next one.

(Refer Slide Time 12:50)



Now, here, the summary of the previous slide was that we could understand this serial-in kind of a operation is also, similar to a shift operation because we were shifting the input every time. All the inputs are being shifted. If you see the shift operation, the shift operations are quite popular in binary arithmetic. So, we are switching the gears so, basically, let us quickly understand how the shifting operations typically works in binary arithmetic.

So, let us say, let us talk about shift right operation. So, shift right operation means, let us say I have some number and I am shifting it right. So, shifting it right means this is my least significant bit, this is most significant bit. When I am shifting it right, then it will become 0 1 1 0. What I have done is, whatever was the bit here, I have discarded, the rest of the things, I have shifted in the right direction.

The question here will be that what should come in the empty space here? In the empty space, here, there would be two choices, either we put 0 or we put 1. What is the reason we will put 1? So, there is, since there is no reason, let us put 0. So, now, let us also, see what this number is and so, if you see, this particular number 0 1 1 0, is actually equal to 0, this is 1, this is 2, this has a weight of 4, this has a weight of 8. So, that means this number is 12, this number is 6.

Now, by shifting right, I am able to divide the number by 2. The initial number was 12. Now, after dividing it becomes 6. So, that means by shifting right by one place I am able to divide it by 2. So, will it always work? So, let us say if there was a 1 here which we were dropping, so, instead of 12, it was 13, but because we were shifting right, that 1 would have been dropped. After dividing 13 by 2, it would still be 16. Because whatever we are dropping is actually a remainder. So, in other words, whenever we are dividing by 2, by shifting, if we are doing a division by shifting, then we are taking the floor of that value. Floor of the result.

So, let us take one more example. Let us say, I have one in the most significant place. Now, I am shifting. After shifting it will become 1 1 1 0. Now, further, there is a question, what should be kept here in the empty place. So, whatever was in the right, that we have discarded but in the most significant place there is an empty space, whether there should be 0, or there should be a 1.

Let us say, why it should be 1? Now, consider that this particular number is a negative number. If it is a negative number, then this 1 was not representing the magnitude but it was representing the sign. If I am shifting it right, then the sign of the number cannot change. It has to be the same as the previous sign. So, there is a possibility that there can be 1 here. But in what cases there should be 1?

Whenever number is signed, then I say that whatever is most significant bit, that should be kept here. This particular method of shifting is called Arithmetic shift right. Arithmetic shift right means I am assuming this number is in twos compliment form, it is a negative number. Because it is a negative number, my, it could be positive as well as negative but it is a twos compliment number, because it is a twos compliment number the most significant bit represent sign, if that sign was 1, here we have to keep 1, if sign was 0, here we have to keep 0.

So, let us consider this also, as a twos compliment number, because it was 0 here so, we have kept 0. So, let us look at the numerical numbers also. If we see this 1 1 1 0 0, if I take twos compliment then I will see that this particular number is actually minus 4. And by shifting right, this number becomes minus 2. So, that means here also, this definition is working correct that it is actually divisible by 2, minus 4 divided by 2 is minus 2.

But, there is also, a possibility that this number is not twos compliment number but it is an unsigned number. If it is and unsigned number, then this particular number represents 16 plus 8 plus 4, 16 plus 8 is 24, 24 plus 4 is 28. So, this number represents 28 and then this number should represent 14. So, that means there has to be a 0 here.

So, if it is an unsigned number then we call it Logical shift right. In case of Logical shift right, whatever we are filling here is always 0, but in case of Arithmetic shift right, if the input number has the most significant bit, whatever is the most significant bit, that would be replaced in the, in the shifted number also. So, this was shift right operation, on the other hand there could be a shift left operation also. In shift left operation, we are pushing all the bits in the left direction.

So, let us say 0 0 1 1 0 was the input and then shifting it by one place, it will become 0 1 1 0 0. Now, all of the bits are being shifted here. Now, whatever was the most significant bit, that is being thrown away. And we are always adding 0 in the least significant place. So, this shifting left is essentially a multiplication by 2 operation. So, here, it was 6 and after multiplication by 2 it became 12. So, it is multiplication by 2.

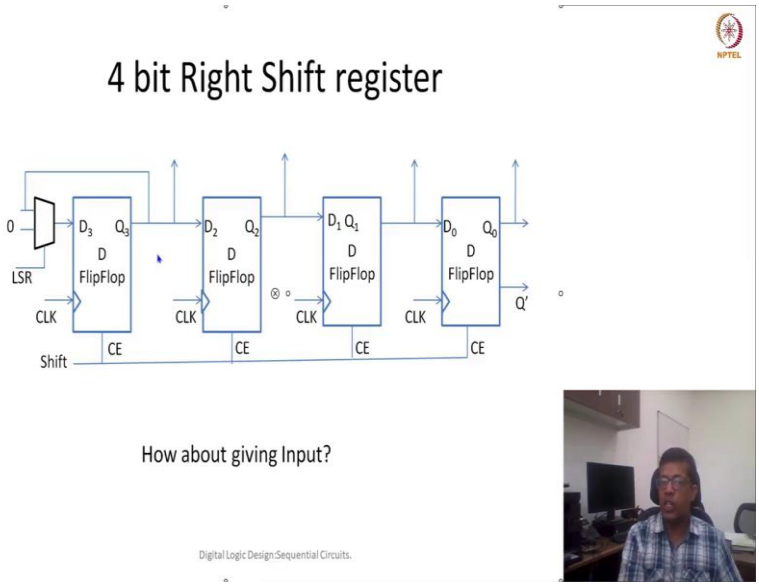
What would happen if there is a one more shift operation? So, if I do one more, then this 12 should become 24, but if it is a twos compliment notation then this could be minus 16 as well. What is the correct value? What should be the correctly, correct thing to represent? Now, because in 32, in this 5 bit numbers, sorry this is minus 8. So, in 5 bits number, the value or the range of numbers could only be from 15 to minus 16.

Because the range, if this particular number, 12 cross 2, 24 is outside the range, this has become an overflow. So, overflow is an error although this number could be a negative number. If we are trying to put it as a positive number, if it was a sign notation then this becomes an overflow. We have to check. Whenever we are shifting left, we have to say that if more significant bit or the sign has changed that means there is an overflow.

So, but if there is no overflow, it will ideally work correctly. So, this is how these shift operations can be done and these shift operations are very important, core part of our digital arithmetic. We have seen that while doing addition, while doing multiplication, these shift operations are quite much required. Both, logical shift operations as well as arithmetic shift operations.

Now, in the previous slide, we have seen that this serial input can be done efficiently by registers where output is cascaded, my output is given as an input in the next flip-flop. So, that means my registers can also, work in a very efficient way, like a shift registers or shift, they can be helpful to implement the shift operations. So, if we have to implement a shift right operation, let us see that how a shift right operation, this shift right operation can be done using shift registers.

(Refer Slide Time 22:31)



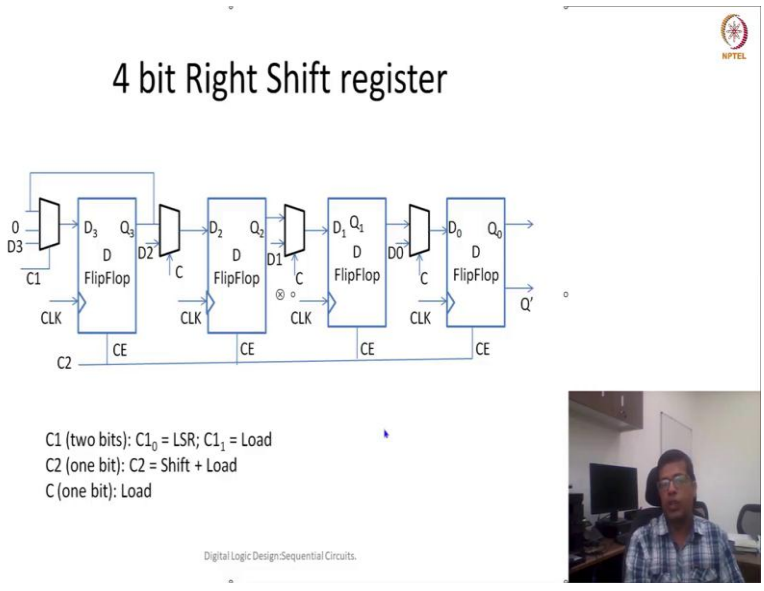
Now, this diagram is similar to the previous serial in shift registers so, here, each output is fed as in input to the next register. Now, this output Q2 is given as an input to D1, Q1 is given as an input to D0. Now, given this kind of a setup, can we use this as a regular shift register? I think, I suppose yes. The only thing we have to make sure that to make arithmetic shift operations correct, we have to make sure that the next, whenever we are shifting, this Q3, because this is the most significant bit, this most significant bit can be given as an input to my most significant register bit.

And then, if we are doing left shift operation, then 0 would be given to, as an input to my most significant bit or the, whatever was the sign bit that would be given as an input to my least significant bit, so, if the most significant bit. So, this LSR, whether it is a logical shift operation, if it is logical shift register, then we have to say 0 would be given otherwise the, whatever is the Q3, that would be given as a, whatever is the most significant bit, that would be given as an input.

Now, the next question is, how about we can give input. So, here, typical shift registers may not be serial input. They would be parallel input, parallel output, the first register which we have studied in today's class. That all the input would be given in parallel. So, outputs are certainly parallel. So, Q3 would be read from here, Q2 would be read from here. Q1 and Q0 will be read from here. But what about the input? So, that means input also, will come in parallel. So, D3 or basically D3, D2, D1, D0, there should be 4 inputs here. So, then, what should we do?

Let us think about it for a while. Maybe, yes. sSo, we may have to have additional multiplexer here. Here, there should be a mask, there should be another input of this mask. At all the input there has to be a mask, either it is coming from the output of this shift register, output of the previous flip-flop or it is coming as an input. Who will decide? If the load is there, if we would like to load the shift register, then we are taking it from the data input. Otherwise, it can work like a shift register. So, let us see in the form of diagram, how does it look like.

(Refer Slide Time 25:27)



So, these, these would be the registers, their multiplexer introduced. Now, when we are introducing this multiplexer here, instead of two inputs, there are three inputs. D3 is given as the input of this mask, similarly D2 is given as a input of this mask but output Q3 is also, given as the another input of this mask. So, you see, D1, input could be either Q2 or it could be D1. Similarly, for all other cases, all these multiplexers are controlled by control input C. Here, this is controlled by control input C1.

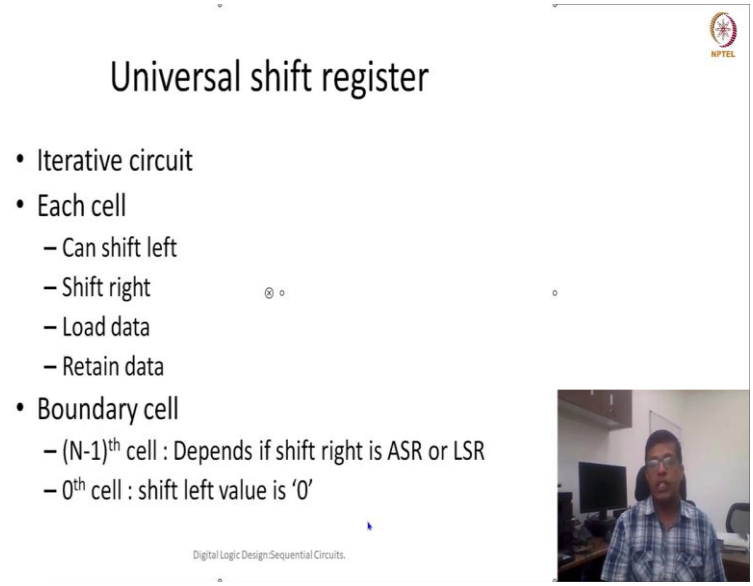
Now, all these chip enable or clock enable are controlled by my clock signal, sorry control signal C2. So, if I see here, because the three input mask, there has to be at least 2 inputs. Input 0 and input 1 means that we are performing some shift operation. While, input 0 1 2, if input 2 is there, so, it, C1 is going to be a 2 bit control and in this two bit control, we will see that control 0 would take Q3 as an input and control 1 will take 0 as an input.

So, that means in both these cases we are shifting. And because this is 0, this is 1 case, so, we can take left shift, even if we are performing left shift operation, then we can take the zeroth bit as that and if we would like to perform a load operation, then we can make a first bit of C1 control as 1. So, that means D3 would be taken as a, D3 would be given to input D3 of this flip-flop.

So, similarly, this C2 control, this C2 control, that means when we have to make this clock enable. This clock enable, or this clock enable would be 1, if there is, either we would like to load or we would like to shift. In either of these cases, this clock enable has to be 1. So, otherwise there is no clock enable. So, that means my register value would remain as such.

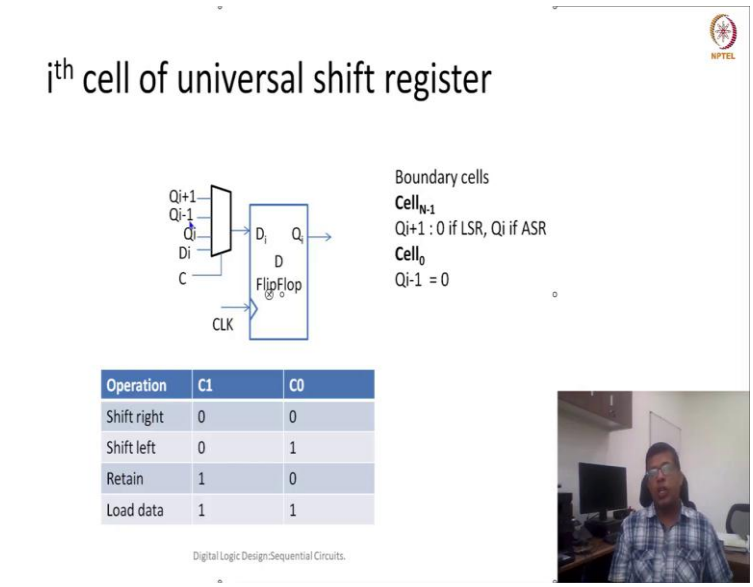
Now, all of these C signals which are deciding whether to take input from previous flip-flop or D2, it would be decided by this control signal. This control signal could be a load 1. So, if load is 0, then it would be taken from this Q3 or if load is, if load is 1, then it is taken from D2. Similarly, for all others. So, this is how we can design a 4 bit shift register which can load the input parallely, as well as which can shift all this bits from right, from left to right.

(Refer Slide Time 28:36)



So, can we design a universal shift register? A universal shift register can shift in left direction as well as shift in right direction, can also, load data, it can also, retain data. So, if we think of it this way and let us think from this particular circuit also, can we extend this to make a universal shift register? So, if we see this circuit, this circuit is essentially iterative in nature. So, basically if we are able to design one particular cell, all other cells can be replicated. The only exception would be boundary cells. Like here in boundary cells we have we have different control structure. So, if that is so, can we design it? So, let us try to see how we can design a universal shift register.

(Refer Slide Time 29:35)



So, in case of a universal shift register, if I would like to shit left, if I would like to shift right, then my input to each of this ith D flip-flop would be coming from Q i plus 1. So, let us say this was D2, then this is coming from Q3. And if it was a shift left register, then this, the next input, one of the input has to come from, if it is a Di then it should come from Di minus 1. So, let us say if it is D2, then it should come from Q1.

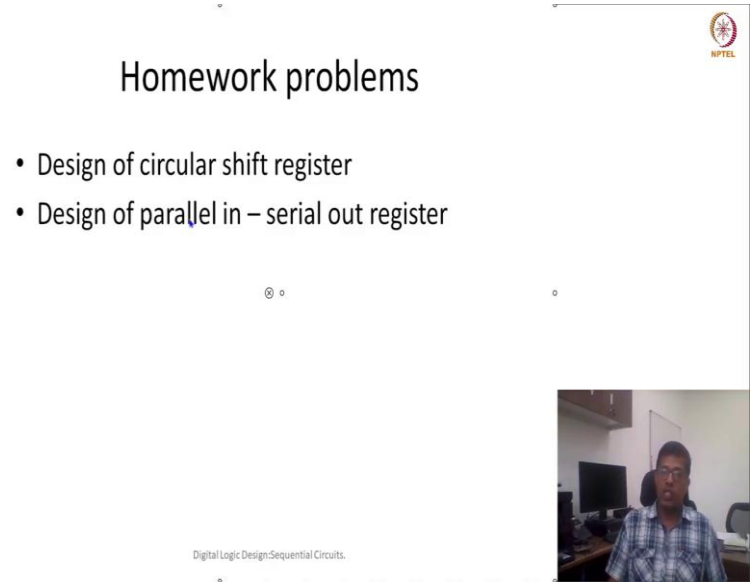
If I would like to retain or this is a case of, we are saying, this is a case of clock enable or we would like to retain, then we can take the input directly from the output of this register itself. And in case, we would like to give parallel input or we would like to load the data, one of the input should also, come from data input, ith of data input and my control will decide.

Because there are four different inputs, my control be a 2 bit structure, C0 and C1 and if it is 0 0, it could be shift right that means the input is Qi plus 1. If it is 0 1, it could be shift left, that means the input is Qi minus 1. So, sorry, 0 0 means Qi plus 1 and 0 1 means shift left and it is Qi minus 1. If the control input is 1 0 so, that means my input is Qi.

In case of 1 1, I would like to load the data so, this Di is going to be my external input. So, this way I can design my generic or iterative circuit for my universal shift register, with the exception to boundary cells. What would happen in the boundary cells? There would be, there would be C, let us say it is an n-bit register. My n minus 1 or the most significant bit or most significant flip-flop, that flip-flop, here, we will say that if it is left shift, so, here, this Qi plus 1 or basically this particular input will come from another multiplexer where it could be 0, if it is left shift register, left shift operation; if it is Qi, it could be Qi, if it is arithmetic shift operation.

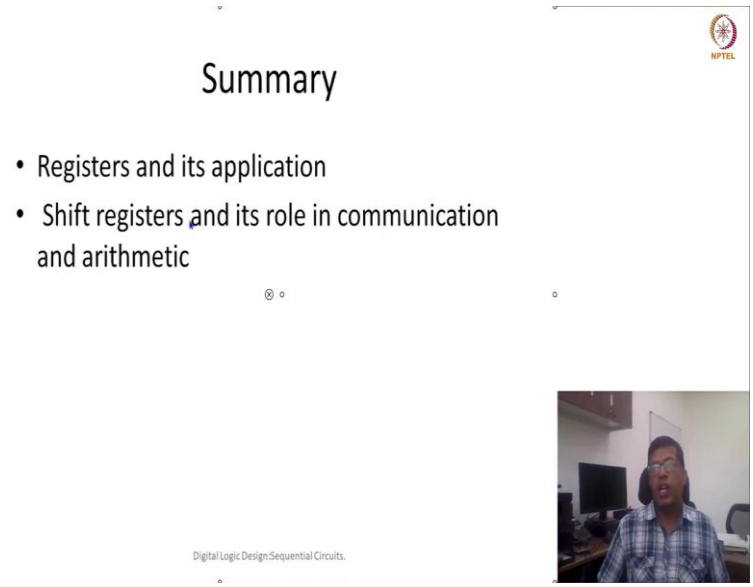
Similarly, in case of cell 0, that will also be a boundary cell, in case of cell 0, my Qi minus 1 would be 0. So, basically here, because there is no Q minus 1. In case of D0, this is going to be Q minus 1, because there is no Q minus 1, so, essentially it would become the, we can directly tie it to 0. So, that way my cell 0 will be a boundary cell, it would have slightly different implementation.

(Refer Slide Time 32:40)



So, this is how we can make a universal shift register and this is, like, if you want to extend it further you can design, maybe a circular shift register. You can also, design a parallel in serial out register. One more similar exercise could be that in Universal shift register there could be of another case where it could be a serial-in parallel-out register as well as parallel-in serial-out register as well as parallel-in parallel-out register plus shift register. So, that will make a completely universal shift register.

(Refer Slide Time 33:18)



So, with this, I would like to summarize this lecture. In this lecture we have seen what is a register, what could be its different applications. And we have also seen a particular case of

shift register which is a shift, we have seen a particular type of register as shift register and we have seen that this shift register could be a quite interesting in doing different kind of arithmetic like addition, sorry, arithmetic like shift by multiplication by 2, division by 2, maybe arithmetically or logically. So, with this, I would like to close. Thank you very much.