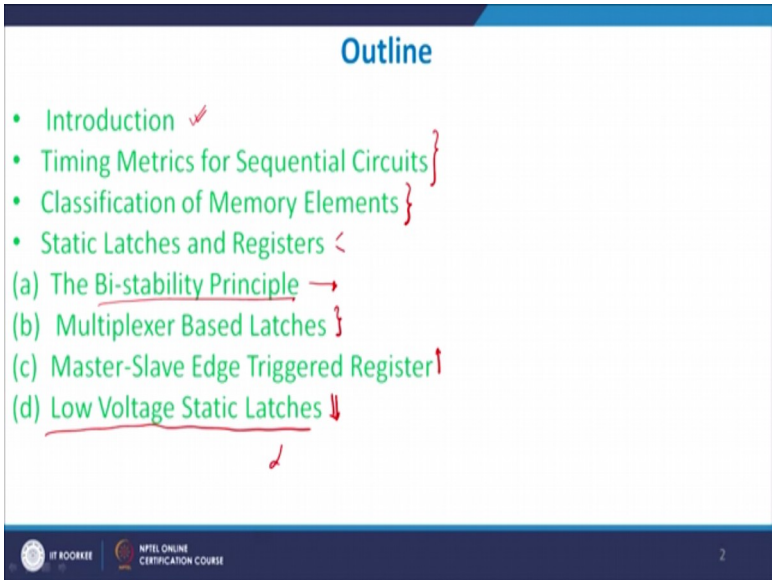
Microelectronics: Devices to Circuits
Professor Sudeb Dasgupta
Department of Electronics & Communication Engineering
Indian Institute of Technology Roorkee
Lecture 57 - Sequential Logic Design- I

Hello everybody and welcome again to the NPTEL online certification course on Microelectronics Devices to Circuits. We will start today with a new topic within the domain of microelectronics and that is design also and that is Sequential Logic Design and this is part one of the sequential logic design.

In our previous modules we had studied about combinational logic, what is the meaning combinational logic and then realization of various gates using combinational logic. We also understood what is the meaning of propagation delay, power area and on which factors this depends. So by this time given therefore a logic Boolean expression, you will be able to design a simple logic diagram, means a simple logic using gates at transistor level. That is at least relevant for the combinational logical block and you should also be therefore able to calculate how much amount of power is dissipated and amount of when it is in dynamic operation and when it is in the static operation. Today, we will start with a totally new section of this design and that is sequential logic design. So let us just have a look at the outline of the course.

(Refer Slide Time: 01:56)



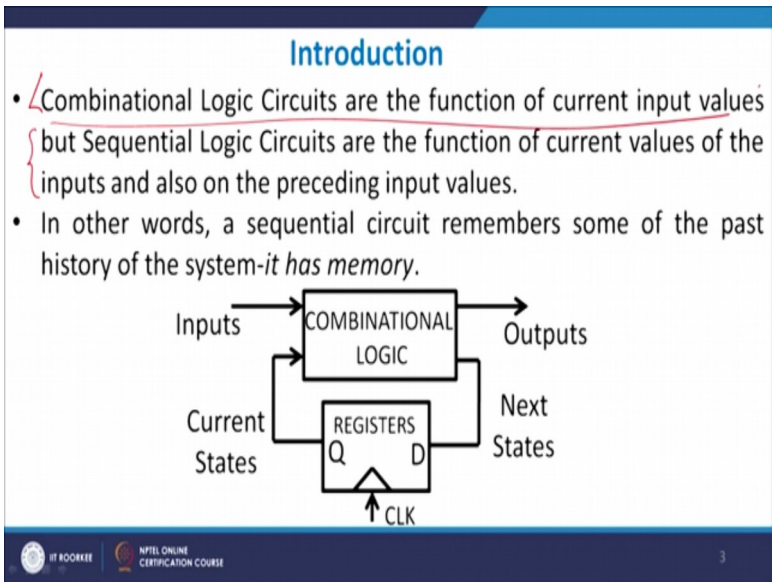
The outline is something like this, we will introduce to you the logic design, we will introduce to you the logic design first of all, what is the meaning of sequential logic, then

very very important term, we will look into the timing metrics of sequential circuits. And we will see that when we are discussing combinational logical blocks, the timing of the clock or the timing of the data was not very sacrosanct or not very important but in a sequential logic the timing at which the data is coming with respect to the clock is a very important perimeter and we would be looking into that in a detailed manner.

Of course, therefore we will explain to you why sequential logic is a memory and therefore the classification of memory elements. We will look into static latches and registers, 2 types of memory. Then we will come to Bi-stable principle, which means that what are the stability points in a sequential design using say for example a simple latch and we will see what is the meaning of bi-stability principle in a latch.

We will also look into mux based which is multiplexer based latch, then Master-Slave edge triggered latch and then finally end our discussion with low voltage static latches. These low voltage static latches is primarily meant for extremely low power technology. So this is for low power, this is Master-Slave is generally meant for those cases where you want to cascade, so this will be a cascaded mode and these are all latches. So mux based latches, and each one us we will, each one of it will be discussed in details regarding its functionality as well as timing requirements.

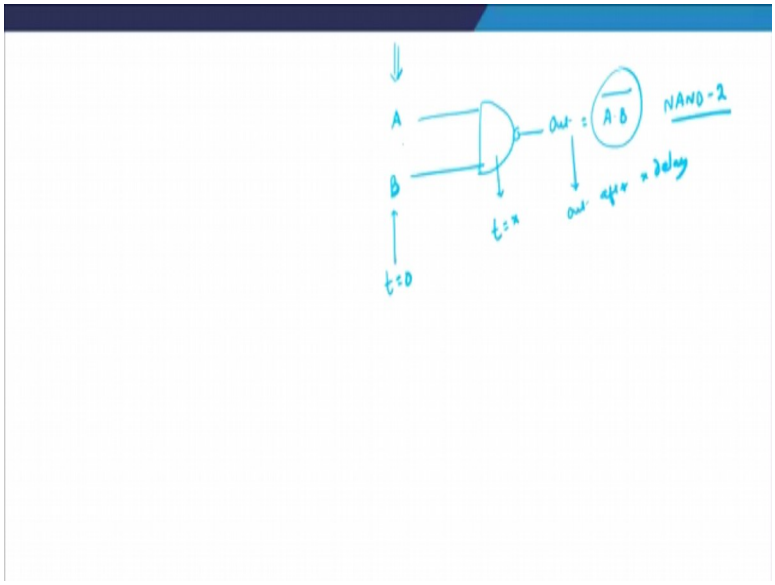
(Refer Slide Time: 03:46)



We have also seen or we will be seeing, we have also seen that if you look at a combinational logical block for example, the as you can see the first bullet here, the combinational logic circuits are the functions of current input values. Let us not, you just do not have a look at

these last 2 lines but just have a look at the first line of the introduction part and you see the combinational logic circuits are function of current input values.

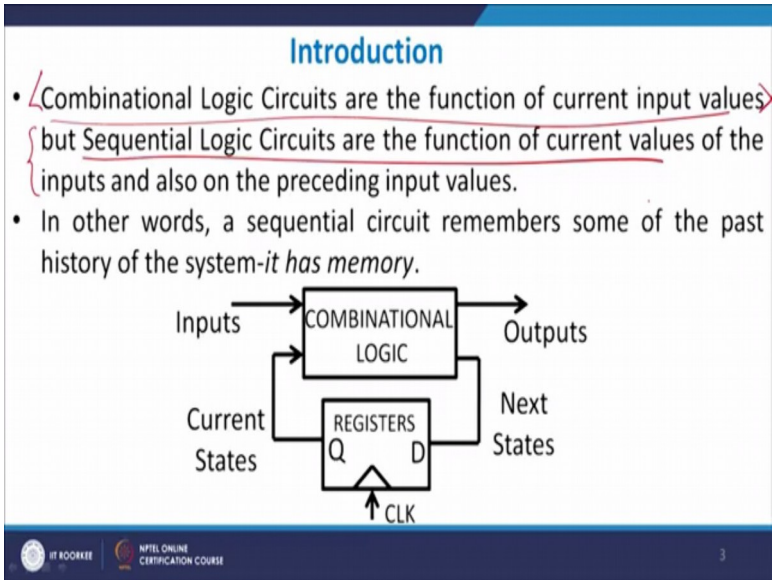
(Refer Slide Time: 04:20)



I will give you an example right? We have already seen this earlier but to give you, just to refresh your memories as far as this is concerned, you can see, understand what I am trying to say. Say for example, you did have a 2 input NAND gate. Now you have 2 inputs suppose and input A and input B and this was output and this was equals to A dot B compliment. So this is a NAND 2 operation which means that output at any point of time which is A dot B compliment will only depend upon A and B which is the input at that particular instance of time.

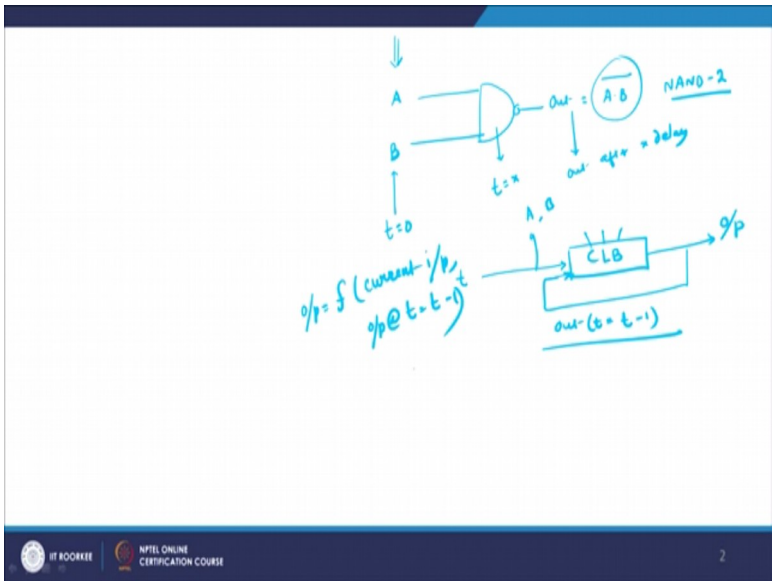
So let us suppose this is happening at t equals to 0 assuming that this is say you have a delay of X then after X millisecond or X nanosecond the output will appear here, that will appear after X delay and this will only depend upon A and B which is true also, it does not depend anything else if you very carefully look. Be it NAND 2 logic, NOR 2 logic, XOR anything it does not depend upon anything else except the primary inputs which are fed into the input side of the NAND gate.

(Refer Slide Time: 05:28)



Same thing cannot be said about sequential logic and that is where it is different but sequential logics are a function of the current values, no doubt just like combination logic but they also depend upon the preceding input values, which means that the output of a sequential logic will depend upon the primary input coming now plus the input available to it at t minus 1th precedence. So everything has happened let us suppose at T, then let us suppose everything is happening at t then in the sequential logical file at t let us suppose I have got a combinational block.

(Refer Slide Time; 06:12)

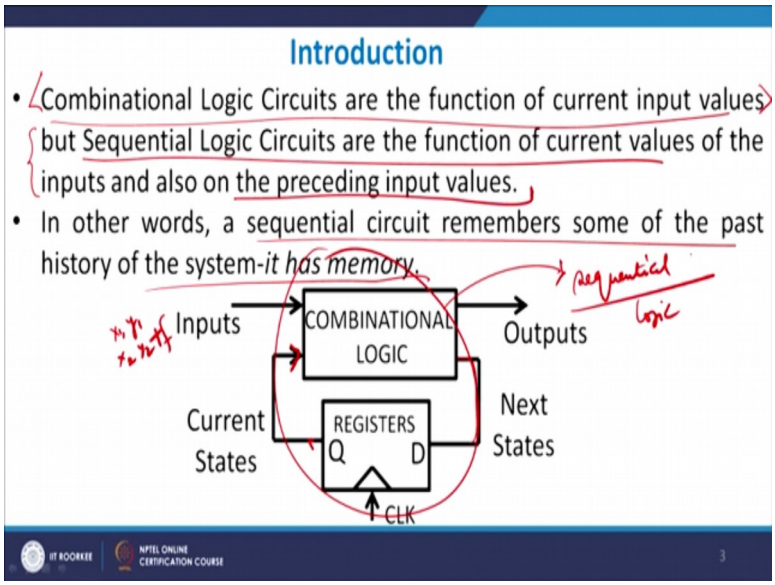


Let us suppose I have got a combinational block C combinational logical, logical block we refined to the CLB and then we get output right but now if this input another input I am

adding here this input is basically let us suppose A and B there are two inputs here and I am adding in the third input which depends upon out of t equals to t minus 1.

Which means the previous state output is not feedback in to the input side and therefore the output now is not only a function of the current inputs but it also depends upon the output at t equals to t minus 1 and that is the basic difference between the sequential logic and a combinational logical block.

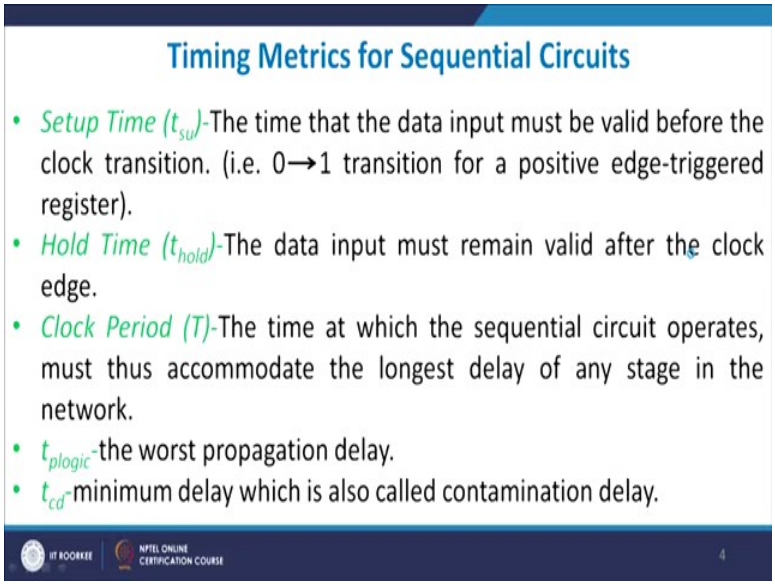
(Refer Slide Time: 06:59)



If you see the same diagram, the sequential circuit remembers some of the past and therefore it has got the memory, as you can see why past? Because you are at least storing some data till the next set of inputs are coming to you. Say you have set some input x and y right, it gets evaluated by combinational logic bar, gets an output and then output is stored in this register here, this is basically a memory element, do not worry about it. And then this are fed into a current value, what current value? The next x.

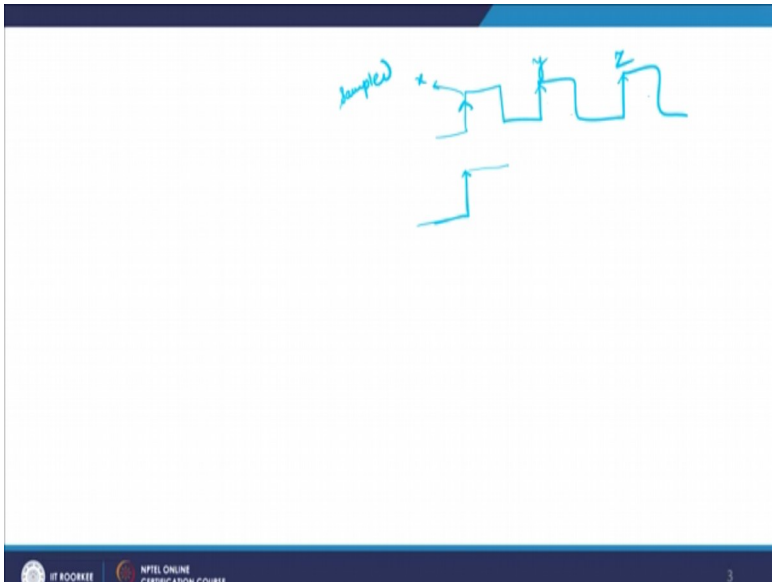
Suppose it is x_1 y_1 and the next is x_2 y_2 but x_2 y_2 plus the feedback from the register will also come which means that this whole logic. So this total is known as sequential logic. This is actually your sequential logic, this whole block of sequential logic will actually store certain bits of information onto its register till a time the next pulse or next input wave train comes. Therefore, output not only depends upon the primary input at that instance of time, it also depends upon the input from the previous state and therefore we also refer to is that sequential logical design also has a memory in it.

(Refer Slide Time: 8:10)



Important points to be remembered, because these are very very important timing diagram metrics for sequential logic and we will go one by one in this case. I will explain to you what do I mean by that, see typically when you have a sequential logic it is clock driven. So what means that…?

(Refer Slide Time: 08:36)

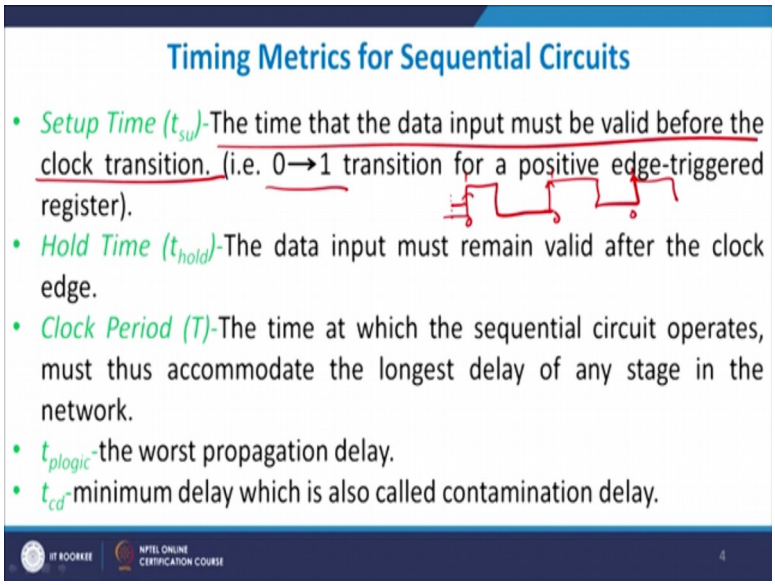


Suppose let us suppose I have a clock here and whenever my clock goes high, my data which is coming from outside gets inserted into the system. So, I will get here data will be inserted, here data will be inserted and here data will be inserted or in technical terms these are referred to as the data will be sampled at these particular points, at x, at y and at z. And so

this is known as and this is therefore known as edge triggered design which means that the data, external data it can only be fed into the system provided the clock is rising.

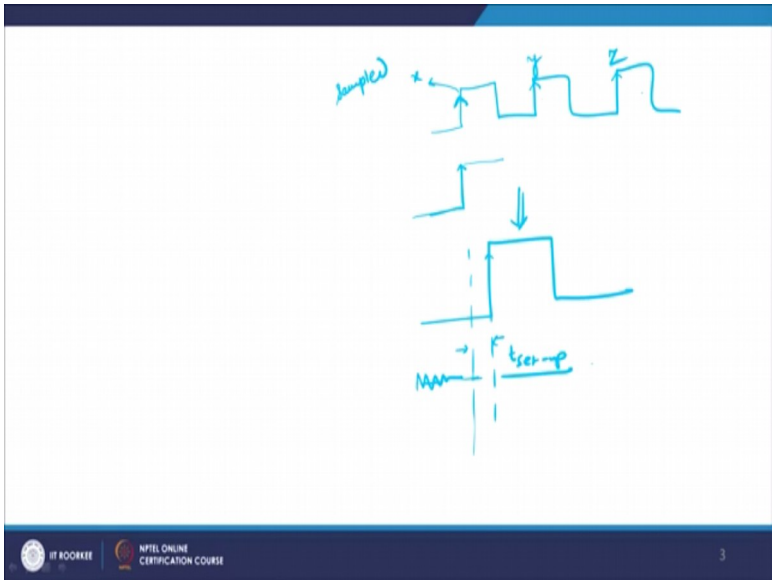
So when the clock is rising and that instant of time you will just catch hold of the data from the outside world. So therefore there are 4 clock cycles, I can catch four sets of data with me. Now the idea, the setup, the concept of setup time is and I will just discuss with you what is the setup time.

(Refer Slid Time: 09:25)



That the time the data input must be valid before the clock transition. So clock will go to 0 to 1 transition as I discussed with you. So when you have edge triggered, positive edge triggered, this is 0 to 1, this is 0 to 1, similarly this is 0 to 1. So these are the all 0 to 1 transitions which is taking place, also referred to as the positive edge-triggered register or positive edge-triggered design. Now you see the setup time is defined as the minimum time as the which the data should remain stable before the rising edge of the clock.

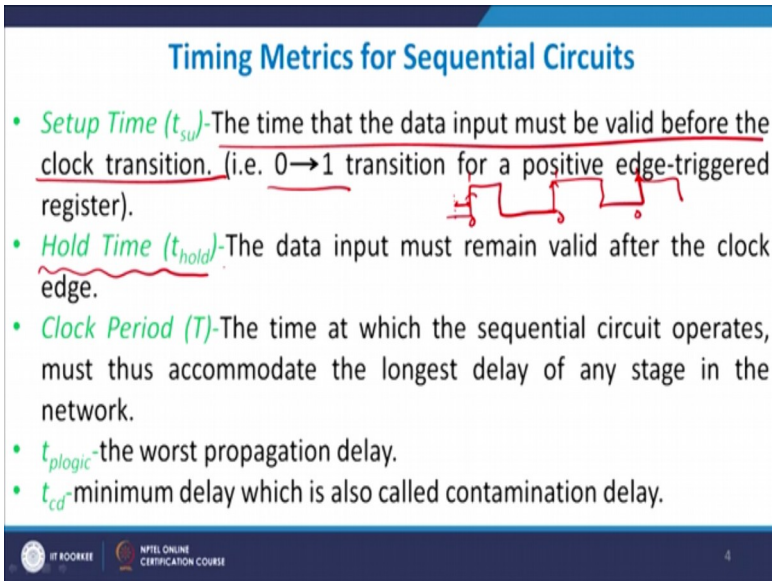
(Refer Slide Time: 10:04)



So, what I am trying to say is that, let us suppose my clock is rising here at this instance of time, right. So your data must be held stable, so your data is moving like this let us suppose, up and down, but minimum it should become stable at this particular point.

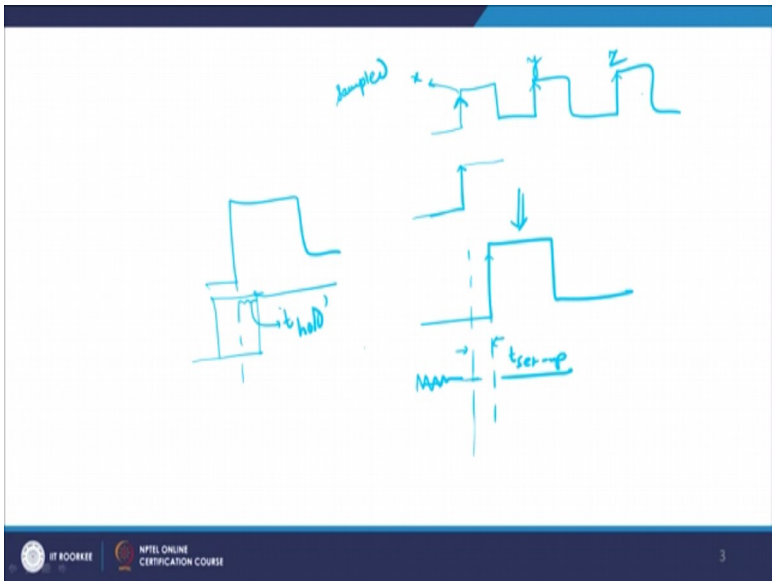
After this, if there is some oscillation in the data from 0 to 1 and 1 to 0, your register or your latch will not be able to evaluate that signal, it will not be able to understand the signal, it will take it as arbitrary noise signal only, you are getting my point? So we define this to be as t_{su}, so what is t_{su}? t_{su} is the minimum time before the rising edge of the clock in which the data should be held stable and that will allow you to evaluate the signal properly or give you it properly.

(Refer Slide Time: 10:48)



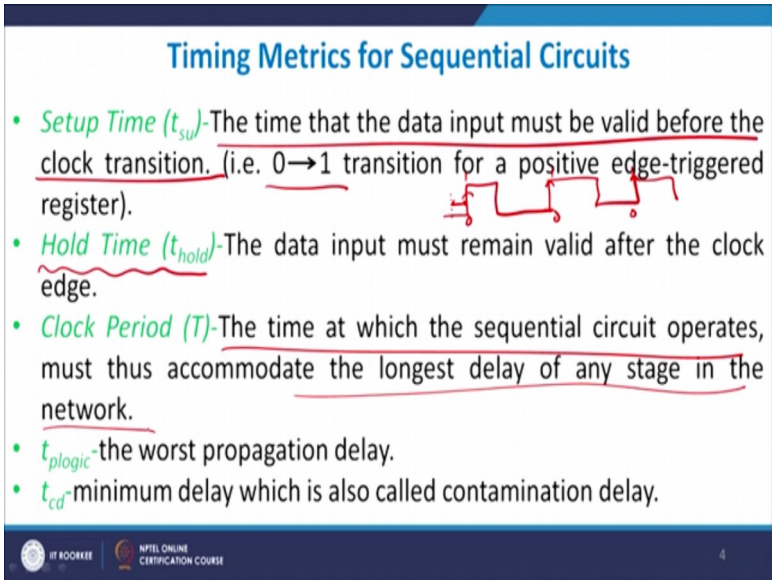
Let me come to the second definition which is known as hold time. Hold time is again very simple, the data input must remain in the same state after the pulse has passed, so setup time is before the pulse has come, arrived and hold time is after the pulse has arrived, you should actually require to have this much.

(Refer Slide Time: 11:09)



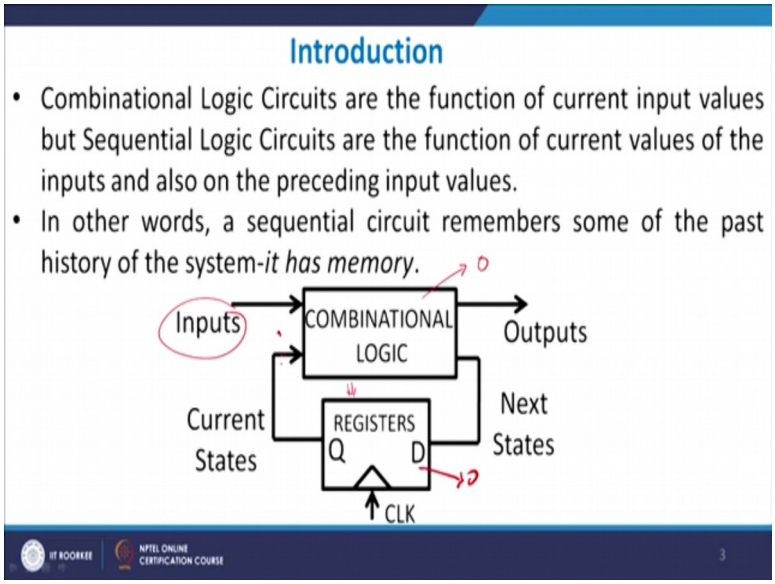
So what is a hold time? If you look diagrammatically, it looks like, so this is the rising edge of the clock and then the data should hold at least till this much point, right. It goes high here and then holds up to this much point, this is known as t_{hold} time. Am I correct? So suppose I am legible to you, in the sense that hold time is the minimum time after the passage of the clock when you are able to have the data stable, after the passage of the clock. And the second time is before the passage of the time and that is what known as hold time.

(Refer Slide Time: 11:45)



Of course, the time at which the sequential circuit operates must thus accommodate the longest delay of any stage in the network. I hope you can understand this point, from the previous diagram you understand.

(Refer Slide Time: 12:05)

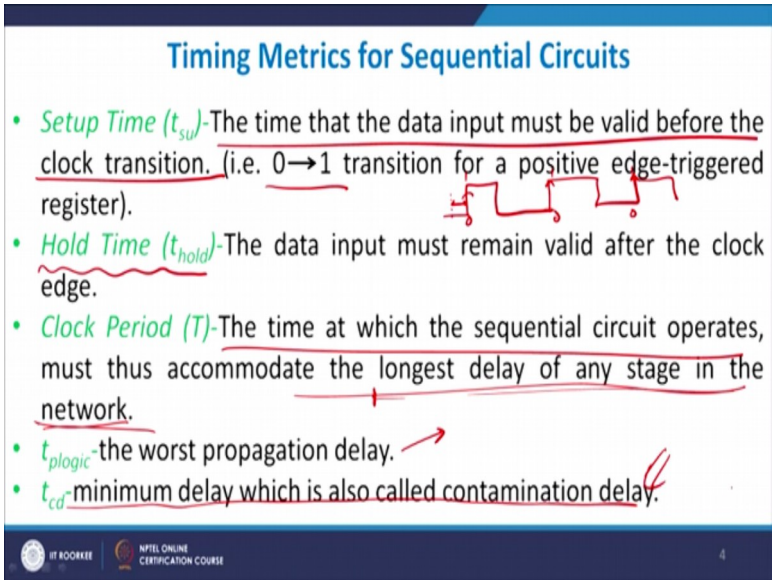


See, if you look here in this slide, you will see that this register has to be minimum delay. Suppose this register would have 0 delay, then what will happen? That, you send an input at this point, at this point you send an input and exactly that say, assuming that this has got 0 as the delay, this has also got 0 as delay, that instantaneously as you insert input here, you will get an output at this particular point.

But understand my basic definition was that the combinational logical box should be able to evaluate the current inputs and the inputs coming from the previous cycle, but, in this case you will find it becoming the same cycle assuming this to be 0 not true therefore. And therefore we define the clock period as the minimum, longest delay, so it should accommodate the longest delay of any stage in the network, this of course you can understand.

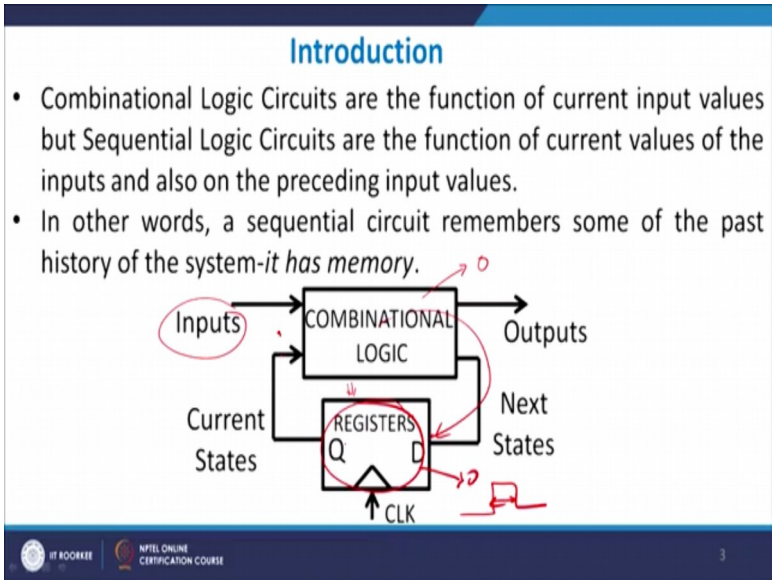
So see, you are driving this clock right, you are driving this clock, this clock is driving this register. So the clock width should be at least that much that it should be able to hold the charge or the data till the whole output is not stabilized, right and that is the basic combinational logical block design is there.

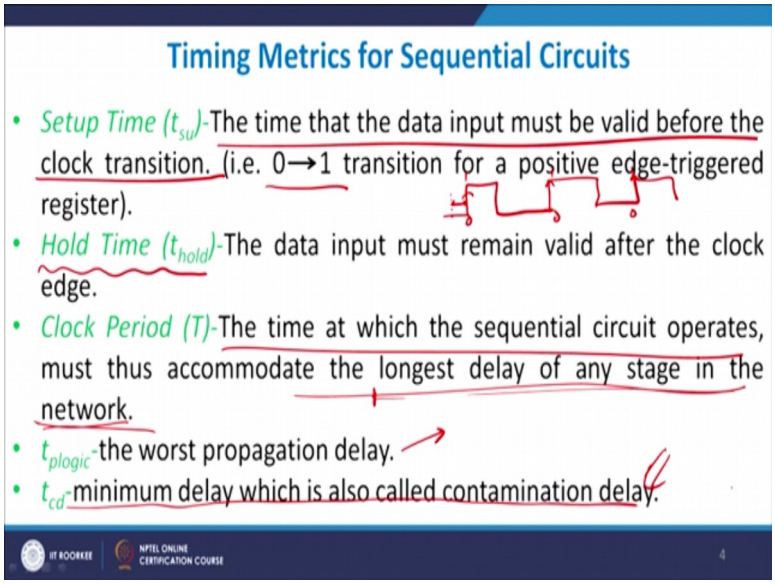
(Refer Slide Time: 13:25)



The worst delay of the logic block is basically referred to as t_{plogic} and the minimum delay is also referred to as t_{cd} and also known as contamination delay. So we have four types of delay: set up time, hold time, we also have clock period which takes care of the generic clock. We have t_{plogic} which is basically the worst case propagation delay across the network, combinational logical block, and t_{cd} is the minimum delay which is also known as contamination delay. t_{cd} delay is also referred to as contamination.

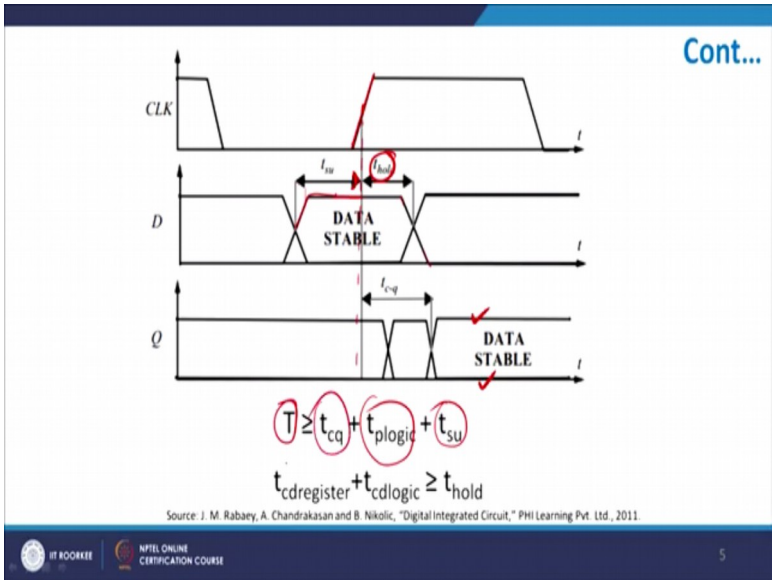
(Refer Slide Time: 13:58)





C to D is basically referred to, if you back the previous slide, C to D is basically this to this delay, right, overall delay. This has to be actually very small or it should be very small. Contamination delay should be 1. So if you want the sequential logic to move fast, you need to reduce these delays drastically.

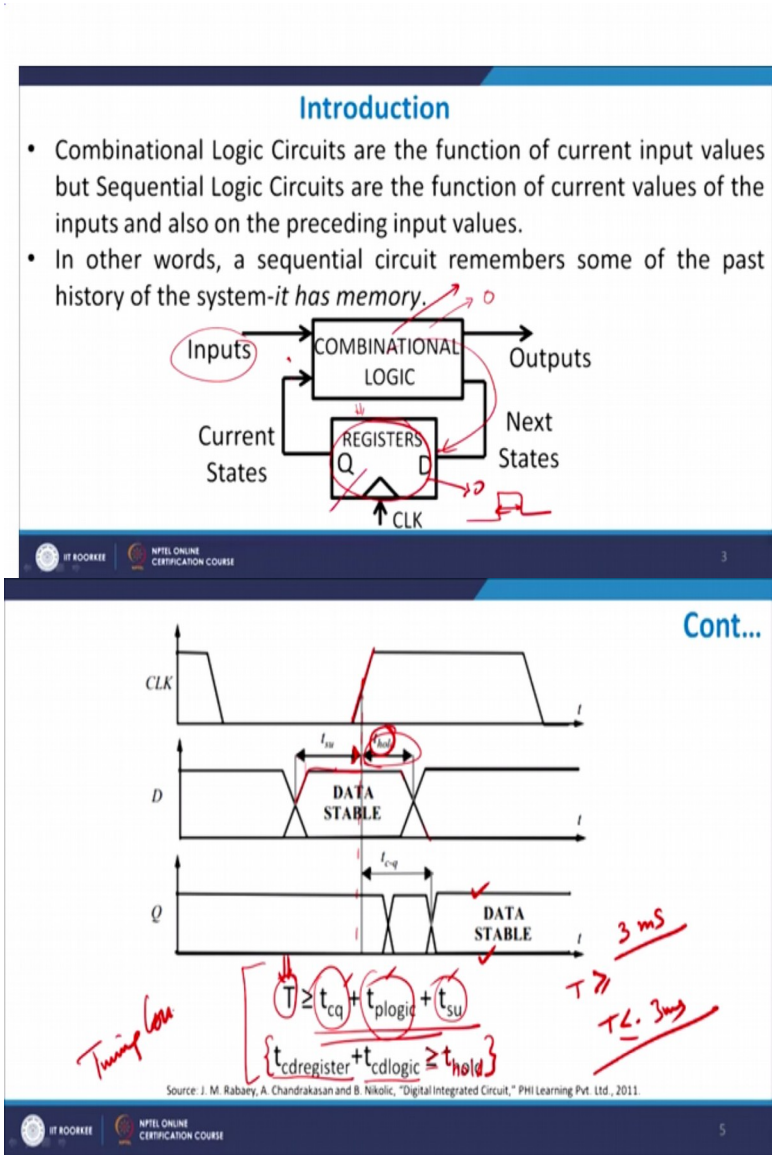
(Refer Slide Time: 14:14)



The same which I discussed with you just now has been shown diagrammatically here. So you see setup time is data, so the data is stable here. So this is the rising edge of the clock, let us suppose. This is the clock, right, so clock is rising edge assuming that it has got finite rise and fall time. So around 50 percent if I take, then we define this to be as a setup time that the time taken for the data to remain stable and we define this to be as t_{hold} because that is the minimum time at which the data has to hold for it can be properly evaluated. What is t_{cq}? t_{cq}, as I have discussed with you is basically the logic block delay and therefore you see the output has actually fallen down here and you are not able to hold it and that is basically my t_{cq}.

After this the data has again held to be stable, why? Because no combinational block, nothing is happening, the data is remaining stable. So we say that the capital t which is the clock of the whole system should be at least greater than t_{setup}, plus t_p logic, plus t_{cq}, right and why is it important, what is t_{cq}? What is t_p logic?

(Refer Slide Time: 15:24)

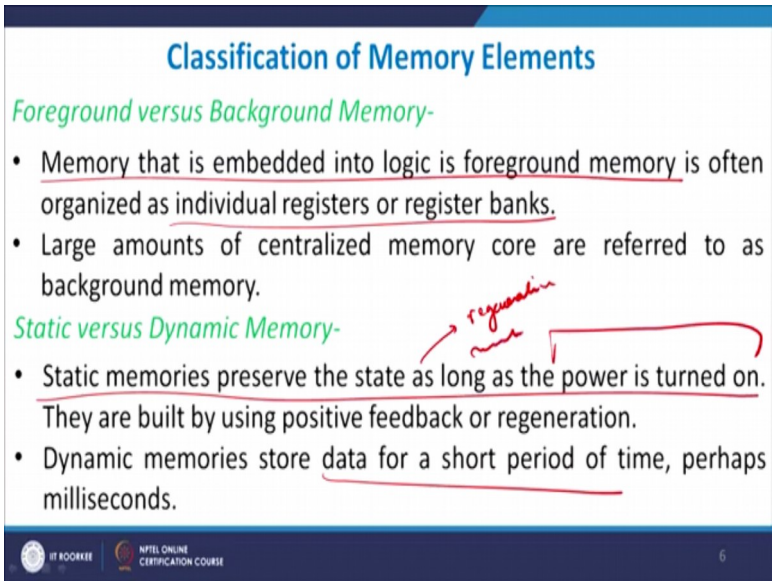


The combinational logical block here, this delay plus this delay, right plus the setup time here is defined as my combinational logic as the total, as means….it is basically your stability criteria and the minimum time should be actually the sum of all these 3. That means if you say, say this is 1, 1, 1, so let us suppose 3 millisecond. If t is greater than equal to 3 millisecond, no issues, if t is less than or equal to 3 millisecond then there is a problem.

The problem is that means you are not allowing at least one function to take care of your sequential logic and therefore it will not remain as a sequential logic, right. That is the one thing, then we define the another set of timing constrains here and that is t_{cd} registered plus t_{cd} logic should be greater than equals to t_{hold} here, what is t_{hold}? t_{hold} is this, remember? So your data must hold, till how much time it should hold? The t_{cd} logic, the combinational logical block delay plus the register.

The register itself should be so, should at least hold that much time so that it is greater than the t_{hold}. Say it is less than t_{hold} then what will happen? Your data is held stable but then you are already ready to give the next set of data. So, you combinational logical block will not be able to understand which data you are talking about in reality and that is the reason this are the two timing constrains which are very very important in the sequential logical design, timing constrains, right and there are two timing constrains here.

(Refer Slide Time: 17:09)



Let us look at the classification of memory elements, the classifications are that you have a foreground and background memory. The memory that is embedded into the logic is defined as the foreground memory, right and is often individual registers or register bank. So in a register you will have CLBs, so these combinational logical blocks acts as a memory array in reality.

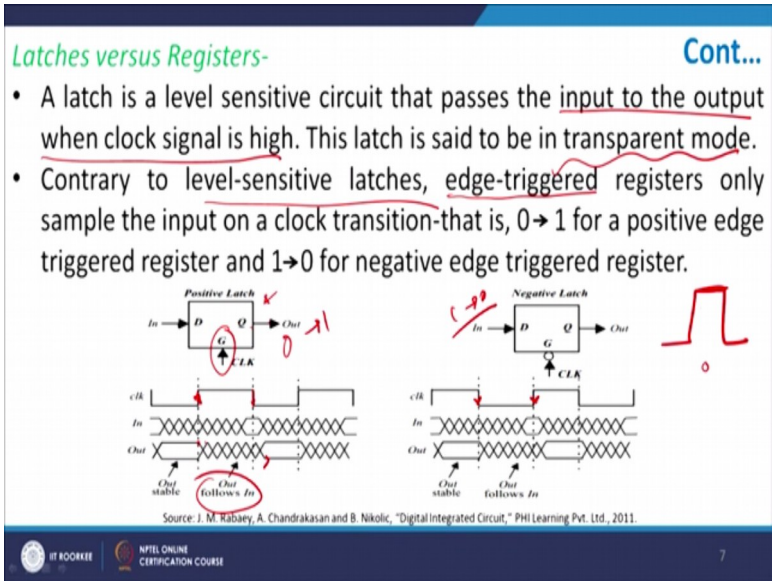
Whereas the background memory is your SRAM, your RAMs which you generally have and these store memory for a longer duration of time. Whereas the memory which is embedded onto the logic, the logic being embedded into the memory in the foreground memory and it is relatively a very low power, right.

Most of the time, whenever we do all sorts of design problems and sequential we use the foreground memory but whenever you want to store a data for a longer duration of time, we use the background memory, right and that is the general rule of thumb which we follow. Now static versus dynamic memory, as I discussed with you static memory are those memories which hold data for as long as the power is on, right, static memory is preserve the

state as long as the power is on and dynamic memories only do for a short period of time perhaps milliseconds. For example, the example might be dynamic RAM, DRAMs, so it is stored in its capacitance small charge but for a few millisecond and then again it will refresh it to let it store for a longer duration of time.

Whereas in a static RAM they preserve as long as the power is on, right and so as long as the power is on, then I can do a positive feedback too. So even if the voltage goes down because of charge getting reduced, I can do a positive feedback network and add charge to it to maintain the level, right. Therefore this is also known as a regenerative network memory. This is also referred to as regenerative memory.

(Refer Slide Time: 19:02)



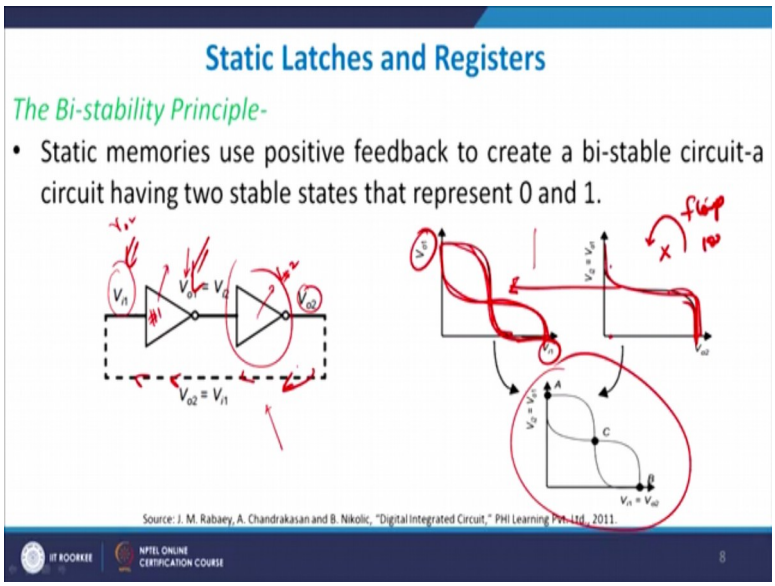
So with this knowledge let me explain to you the difference between latch and register, you must be knowing also, latch is a level sensitive circuit and passes the inputs to the output when the clock signal is high. So this is my latch, so whenever at this state when the it is level, when it is level sensitive and high level it is reached, then only the latch becomes transparent and is able to perform any operation and let the input go to the output with the clock is high.

This latch is transparent latch as I discussed with you. Contrary to level sensitive latches, how I define edge sensitive or edge-triggered registers? With sample input only when during the rising edge of the clock, 0 to 1 positive edge-triggered. When you sample it 1 to 0 then we define it to be as a negative edge-triggered register.

So if you look at this block diagram here, this is basically your positive edge-triggered and on the D and Q, D is the input and Q is the output and this is my clock here. So you see when the clock is going from 0 to 1 right, output goes 1 and it remains stable. When it goes from 1 to 0, the voltage also falls down to follow and therefore out follows in this case by certain delay. Whereas in this case where you have a negative latch, negative latch basically means during the negative 0 to 1 transition.

So this is 0 to 1 transition of the clock, this is 1 to 0 transition of the clock, you will have negative edge-triggered design and that is how it follows. It is almost the same as you look here, only thing is that during the negative edge of the trigger you will have following edges with you and this is positive edges rising here and you will have the output being latched. So when the positive goes high the data is being replicated here, when it goes low the data is left and that is what is a difference between a latch and a register.

(Refer Slide Time: 21:00)



Let me come to a concept of bi-stability principle. Well, what we do is that we have got two static invertors in series to each other as you can see on the left hand side. So if I define V_{i1} as the input voltage to static inverter, number 1 and V_{i2} to number 2 then what will be the output? It will be V_{O1} will be the output of the first invertor which will be exactly equal to V_{i2} for the second invertor and output will be V_{O2}. So V_{O2} now goes inside if it is feedback look and now V_{O2} and therefore if it is V_{O2} here which is just, what is V_{O2}? V_{O2} is just the output of this one, you will feedback into the first invertor. I will show to you that such a pair will only give you two sets of stable points.

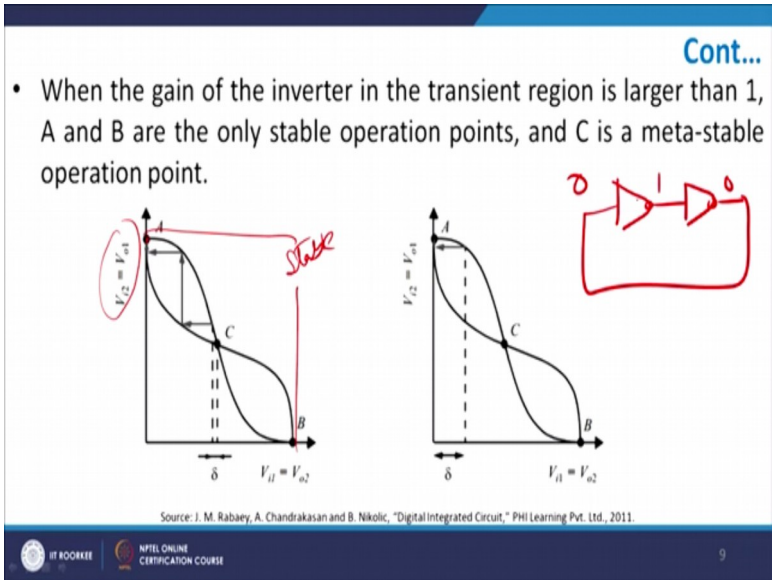
So if you look carefully you forget about this now and you draw the VTC for the first one, you get V_{out} versus V_{in} to be this. So this is V_{i1} here versus V_{O1} which is this one. Therefore, since it is a static invertor it will behave exactly like the VTC of the static invertor which we have already studied when input is low and output is high, when input is high then output is low, fine, we have already done this point.

Now if we go for the second one therefore, you understand the second one's input is the first one's output. So if I want to draw the VTC, I just have to make x to y plane and y to x plane and then the job is done. Once you do that, I get this into consideration right, I get V_{O2} here and it is V_{I2}. What is V_{I2}? V_{I2} is the input to the second transistor right and when it is low, the output is high and when the input is high the output is low, it is very straight forward and simple.

Now what you do is you just flip this flip this by 180 degree right, flip it by 180 degree and place each axis over other, then you will get a figure something like this. You are getting my point? So this is your one, you have another one which is something just like this, so just you just place one over the other. You will get something like this. So this is I am just shifting it on this side right? And this is what you will get.

This is a bi-stable circuit for a memory, I will explain to you where the bi-stability comes into picture but I suppose now you appreciate the point that based on the two stable states and there is one unstable state, using positive feedback I can actually achieve. So this is the feedback loop which I am drawing to, giving, again I will be able to achieve certain important things.

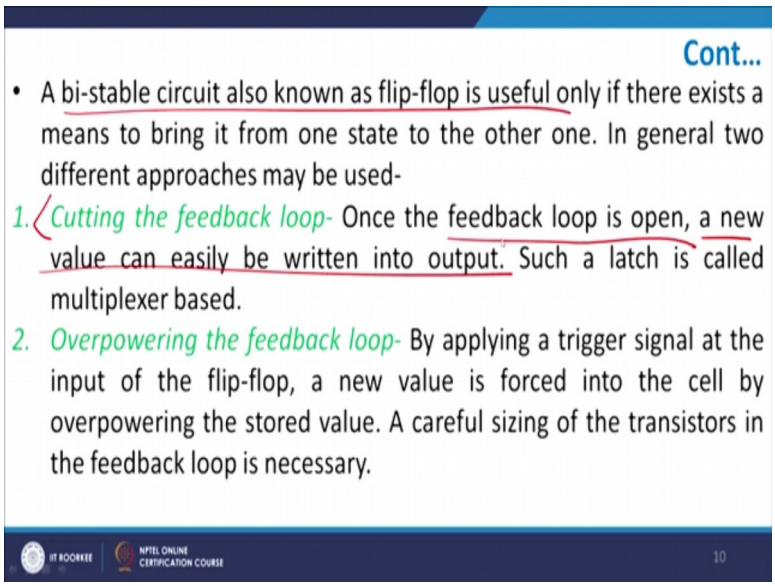
(Refer Slide Time: 23:52)



Now if you look at this VTC, then you see this point A and point B are the most stable points, these are the most stable points and the reason being for example, look at the point B, point B is basically Vinput equals to Voutput2 because you feedback directly from the system and this corresponds to V_{i2} equals to V_{O1}.

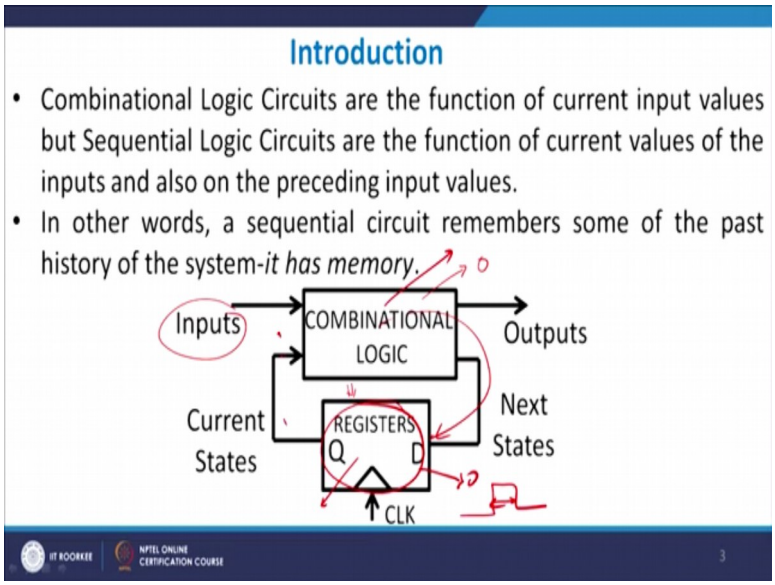
So now if you just place it in A or B, it will always remain stable because you can see very clearly that if you have something like this then if it is 0 1 0, then 0 is feedback into the input side and this is the most stable configuration of two CMOS inverter connected back to back and that is what I am trying to show you in front of you.

(Refer Slide Time: 24:39)



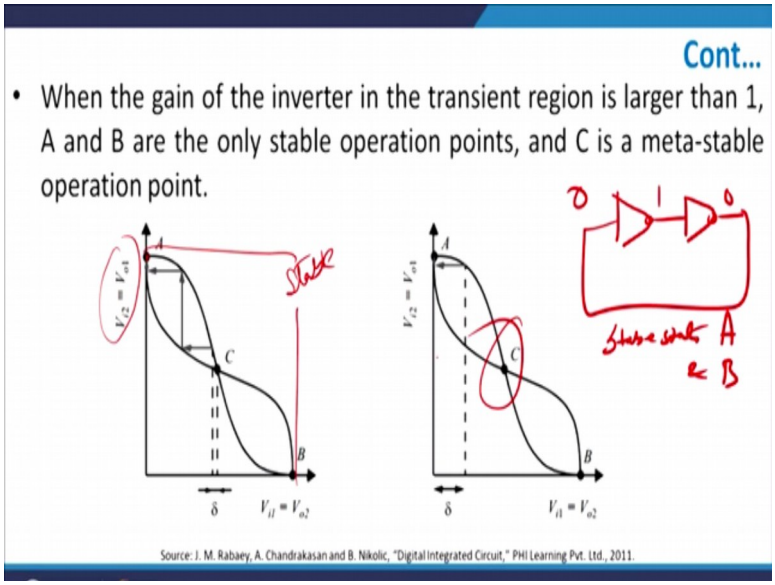
Now with this knowledge a bi-stable circuit can also be known as flip-flop, why flip-flop? Because they are two stable states and by doing some manipulation I can move the stable state from A to B and I can also bring it back from B to A right. I will show to you how this is done. There are two basic approaches by which you can achieve a bi-stable circuitry, obviously you should cut your feedback loop and therefore now the feedback loop is open and now you can write very well onto the output.

(Refer Slide Time: 25:27)



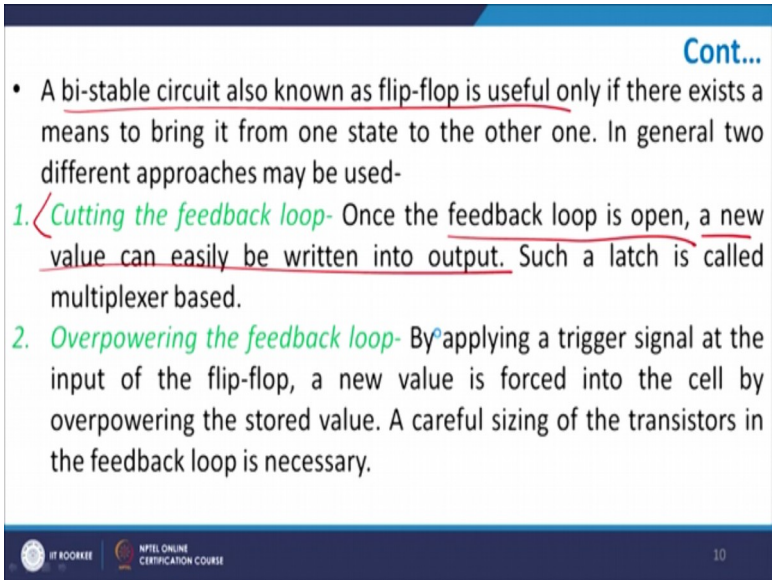
Which means that it is something like, that means, sorry, if you look in this case then if you, for example if you open the feedback loop here, I will just show you from the figure which is there, right. So if you open the feedback loop which is this one, then there is no problem, then it becomes a combinational logical block and as I discussed with you, when you are discussing having a combinational logical block, there is no concept of delay as such and that is the reason you will get a stable state here.

(Refer Slide Time: 25:40)



You will have a stable state at A, so stable states are what? Stable states are A and B, any other state is sort of an unstable state, the most unstable state is basically the C where you have got two VTCs crossing each other, why crossing? Because input of two is exactly goes to the output of one. So I am just overwriting each one over other and I get this into consideration, A and B are the most….

(Refer Slide Time: 26:13)

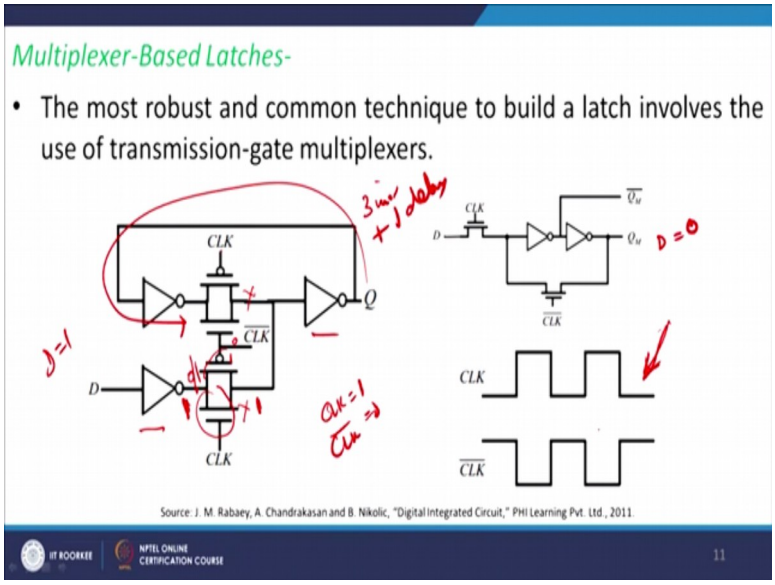


So how do you do it? You when you keep the feedback loop open, no value can be easily written and such a latch is called multiplexer latch, we will discuss this point. We can also overpower the feedback loop by applying a trigger signal at the input of the flip-flop and a new value will be forced on the cell by overpowering the stored value. So if your stored value

is there but I am able to overpower that value by an external source voltage then we can say that your flip-flop or your this thing, the sequential logic is prone to external voltages.

Now you see if you want to make it, another way is that you want to overwrite over the cell within the combinational logical block, then you have to ensure that your feedback loop is relatively very weak. If it is weak it will not be able to push around its value onto the basic combinational logical block. So if you want a combinational logical block to sustain, this is a particularly good idea people use across the research community.

(Refer Slide Time: 27:21)



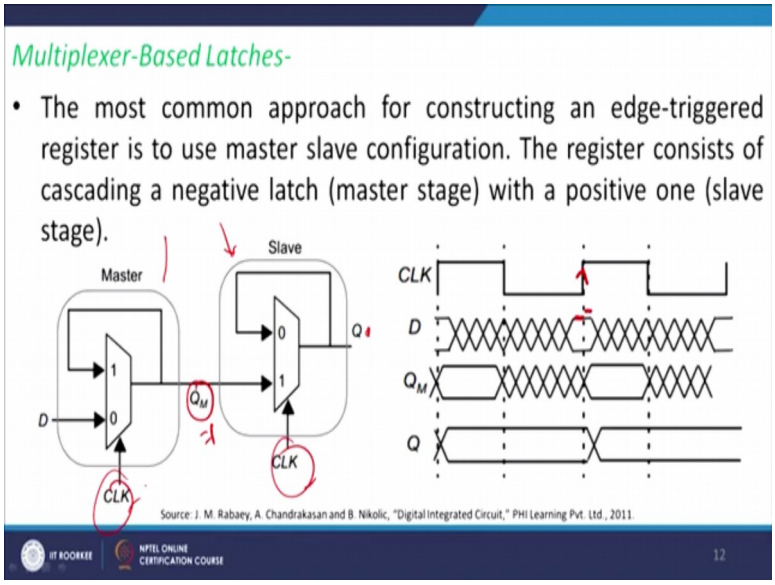
Let me come to a multiplexer based latches, the most popular one is this one and as you can see here, this is basically a Q and so when D equals to 1, suppose invertor this is 0, now let us suppose clock equals to 1 and therefore clock bar equals to 0. So this is 0 and this is 1. So when as I discussed with you in TG when this is 1 this is cut-off, this is cut-off and nothing happens. Now what happens is that suppose D equals to 0 and therefore output of this equals to 1, this makes slightly difficult easier for us that this will therefore be, if it is 1 then this clock will be switched on and similarly this clock bar so if it is 1 the output will be 0, sorry output will be, so if it is 1 if clock equals to high, I we would expect to see a 1 at this particular point.

But please understand in this case the total delay between the D and Q is the sum of this inverter, this inverter and this inverter, why? Once you set the data it passes through this, this but this is not making a difference and then this, so it goes via this, this and then backs to this point. So, I have got three inverter delays, plus 1 this delay because of the clock because of

the two transmission gates or maybe one transmission gate. Now this gives you an idea that having transmission gate reduces your overall on current but the variation of its resistance with respect to input voltage is almost constant and we know that, that is the reason we use a TG design in most of the cases.

This is the non-overlapping clock I was talking about because this is clock and this is clock bar and you will be able to sustain a non-overlapping clock for a longer duration time once you are able to design these muxes.

(Refer Slide Time: 29:16)



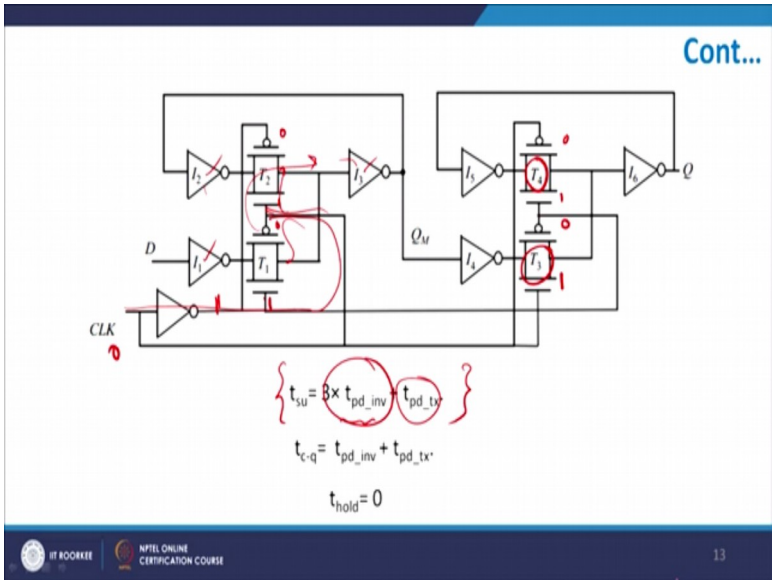
This is another mux based latch, the most common consideration using an edge-triggered, so I have a master here and I have a slave here right .So when the master comes Q1 is equals to 1, it gives 1 here. Whenever the clock is high the slave accepts a data from Q, compares with 0 and 1 and then processing starts place.

So you see this is my clock here. When the clock goes, so this is basically positive high which means that if the clock goes high let us suppose, the data has to be held stable before the rising edge of the clock and it should be minimum held stable till this much point after the passing edge of the clock. And therefore we should be very careful about how we are clocking the whole thing. The second thing is, as I have discussed with, you are applying clock at this stage and at this stage.

Please ensure that these clocks if you are doing it in a lab or somewhere please ensure that this clock and this clock are exactly of the same dimensions giving you exactly same peak to

peak voltage, otherwise you will not be able to achieve such a good master slave configuration of laches.

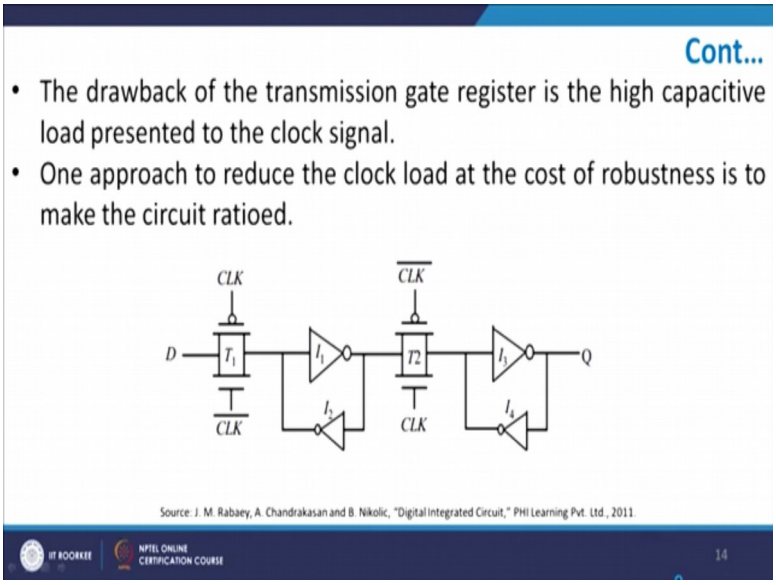
(Refer Slide Time: 30:30)



This is another example which you see and that if you look very carefully here when clock goes say 0, this is 1, this is 1 means this is 1. When this is 1 means this is 1, this is 0 and therefore this is 0. So what happens is that if this is 1 this goes to suppose here right and let me just try the same thing here 1 0 1 0.

So whenever the clock one goes and tries to overwrite here, since it is fed to the PMOS terminal, T_3 does not react at all but T_4 shows a change drastically. So the overall delay will be actually governed by the T_4 delay in a combinational logical block. The total setup time therefore is three times inverter delay, why three times? Because there is a 1, 2 and 3 because the data has to travel from here, it has to go here and then either it should come to this point or come to this point and then finally to the output side. And therefore your total delay is basically 3 pd (propagation delay) inverse, plus 1 transmission gate, so either of the three transmission gates will be required finding out the total delay. We define t_{cq} as equals to t_{pd} inverse plus t_{pd} transmission, if you add those two together I get t_{cq} as one of the important points. Hold time is 0 in this case for all practical purposes.

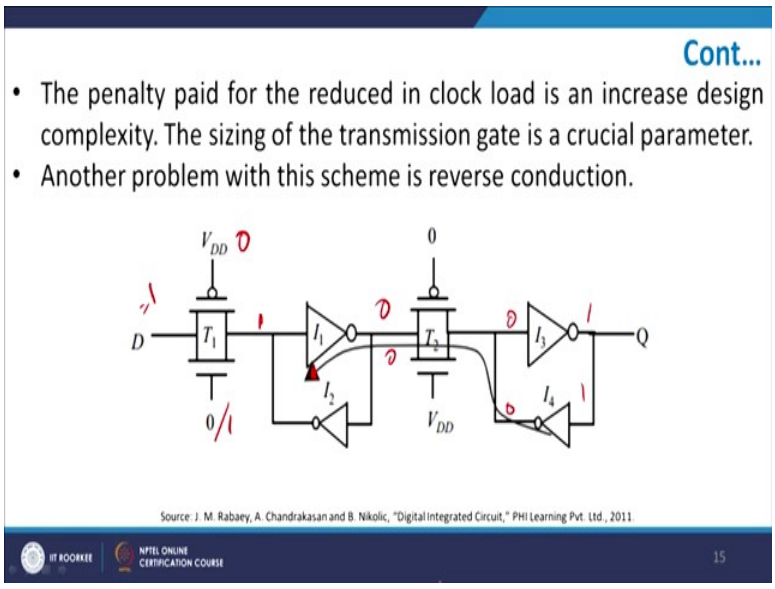
(Refer Slide Time: 31:50)



Now the drawback of transmission gate register is the high capacitive load, we all know this point, because you remember this transmission gate obviously terminates with both PMOS and NMOS on the same substrate and therefore you will have excessive loading of the substrate onto the device itself and as a result you will have large amount of capacitive delay.

Another methodology which people have approached is to make it ratioed logic, remember till now we were discussing that these are all non-ratioed logic and we will not be requiring any ratioed logic in this case but we do require ratioed logic because of the fact that we want to reduce the clock load.

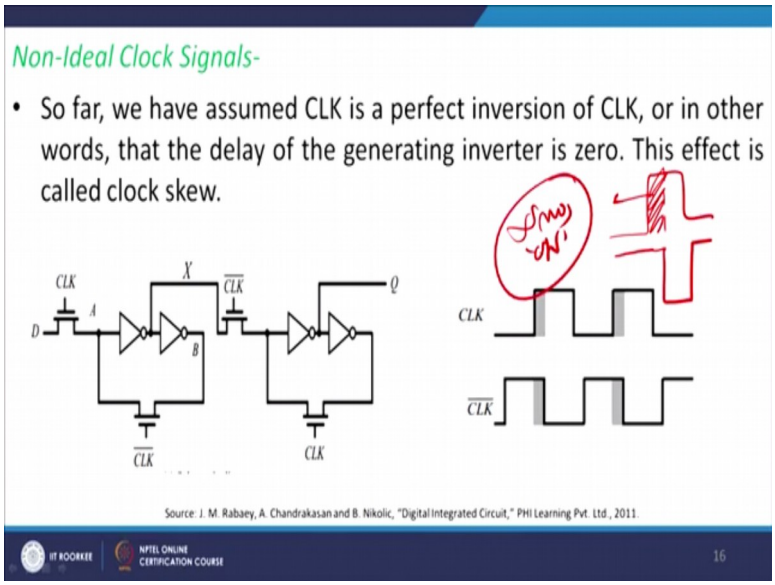
(Refer Slide Time: 32:35)



However, if you want to reduce the clock load, the penalty you want to pay is basically design complexity right and therefore the size of the TG is quite important because I will give you an example. If the TG size is so small that the resistive elements goes very high on chip then you will end up having a very high delay in the output side. Similarly, if these are shorted, you will never have large amount of current to drive your inputs and therefore proper sizing of transmission gate is a crucial parameter.

Now if you see D equals to 1 and V_{DD} equals to 0, so this is 0, this becomes 1 let us suppose. It transmits 1 here, goes to 0 here and then comes to 0 and this is 1, this is 1 and this is 0. And so we get back 0. So I gave 0 and I got back as 0 but the only thing is that now I have a transmission your gate based design which takes care of it. The reverse conduction we need to stop it, so make the area large and we can stop the reverse conduction.

(Refer Slide Time: 33:32)

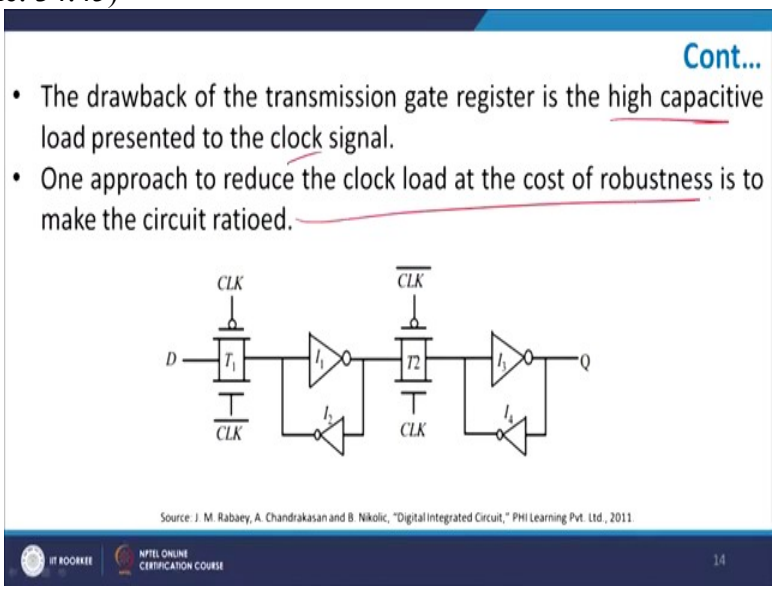


This is how you generate a non-ideal clock signal right. However there will be some delay between two clocks right and that is known as Q delay or clocks skew and this is the diagram of the schematic of the non-overlap clock generation, you can use this as a plug and play sort of option.

What is the problem? This option means this delay means if there if let us suppose your clock and clock bar are not having perfect overlap, then what will happen is then when the clock is high, your clock bar is also clock bar which was supposed to be 180 degree out of phase will be shifted by this much amount and it will go like this.

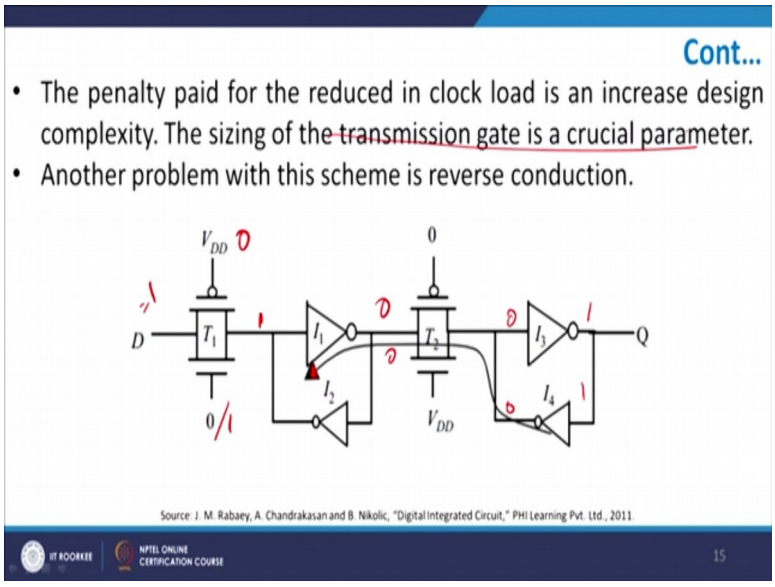
So till this time your NMOS will be still on and you will be actually sampling data even. You do not want the sample to take place but actually it is happening because of the overlap, this is known as a 1 1 overlap in a combinational logical block, and therefore it is also known as a reverse transmission.

(Refer Slide Time: 34:45)



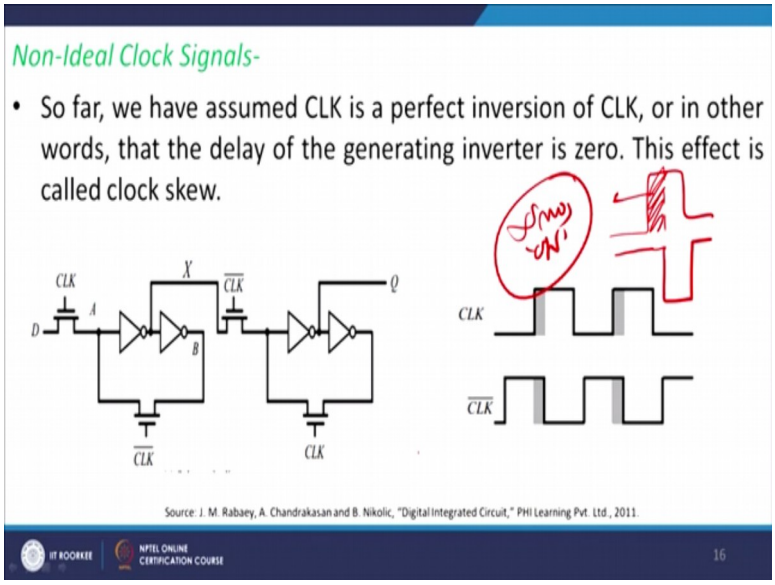
The drawback of, as I discussed with you is the high capacitive load and robustness of the circuit is an important issue.

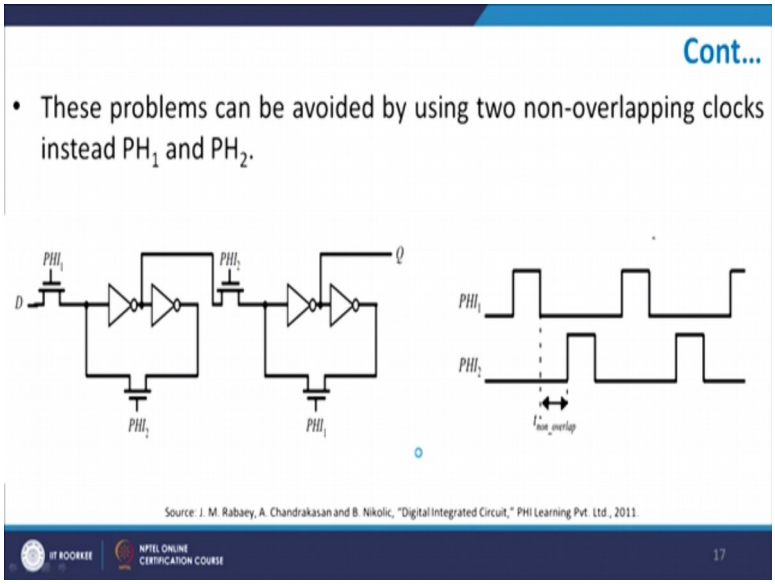
(Refer Slide Time: 34:50)



We have also looked to the penalty of paying, if you want reduce the clock gate and the sizing of the transmission gate is quite important. Larger the size lower will be the loading effect and that is generally taken care of across the board.

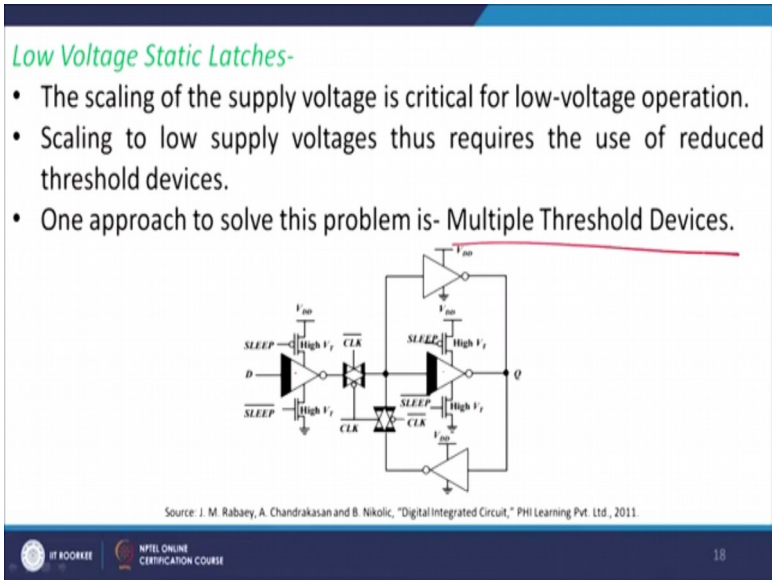
(Refer Slide Time: 35:04)





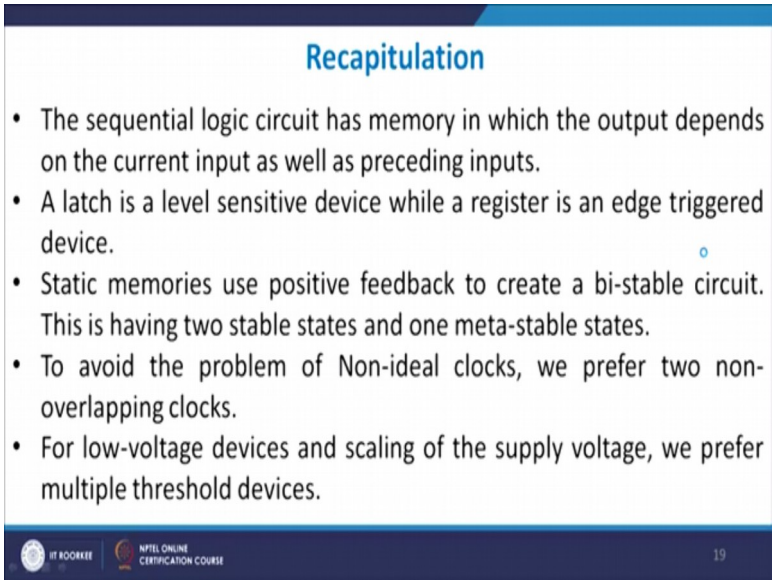
So as I discussed with you non-ideal signal will be those signals which the clock is a non-perfecting clock and this can be done by using two overlapping clocks of just pure non-overlap of each other and we should be able to generate the output file.

(Refer Slide Time: 35:22)



Okay, then we come to the last part of our this module and that is basically low voltage static latches. This is critical for low voltage operations. What we do is that reduced threshold voltages are used MOS devices in the critical path so that even when the dynamic participation is high that can be reduced drastically by applying this sleep mode transistors with a reduced threshold voltage of the device. This is also referred to the multiple threshold device we can use. This is a quite interesting one.

(Refer Slide Time: 35:54)



So let me recapitulate what we have done in this small module, we looked into sequential logic design, how it's made, what are the diagrams we have already seen it. A latch is a level

sensitive device whereas edge-triggered, resistor is an edge-triggered device. Static memory used as a feedback can create bi-stable element, DRAM is one of the examples.

To avoid the problem of non-ideal clocks, we refer two non-planner overlapping clocks, this is what I was trying to say till yesterday. For low voltage devices, the scaling of supply voltage we prefer multiple threshold voltage which means that when we go for low voltage domain, we require multiple threshold devices.

In the critical path try to keep a low voltage devices so that they are switched on and off very easily and the signal can be easily passed from primary input to primary output. So this is what is all about static memory and dynamic memory, we have taken care of the basic ideas of sequential logic. In the next module we will start off again from the sequential logic flow only. Okay. Thank you for your patient hearing.