Switching Circuits and Logic Design

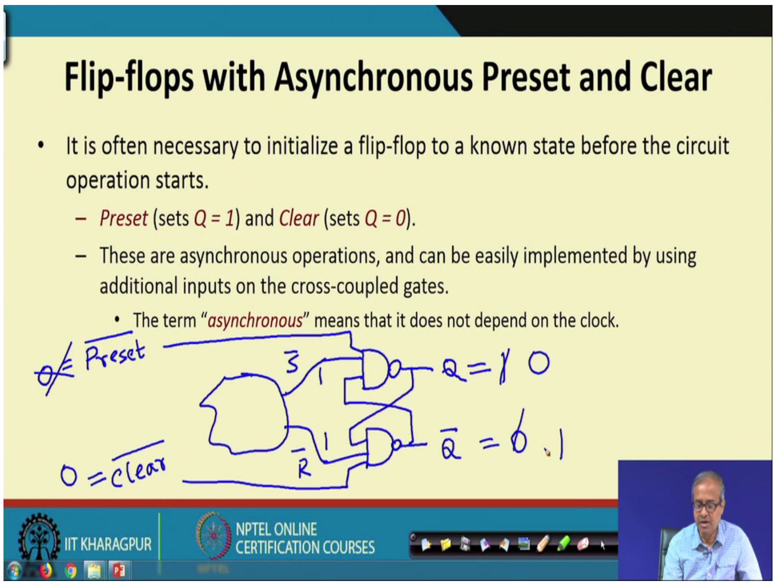
Prof. Indranil Sengupta.
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture - 33
Latches and Flip-Flops (Part III)

So, in this lecture we continue with our discussion on Latches and Flip Flops. This is the third part of it. Now if you recall in our earlier 2 lectures on this topic, we talked about some of the latch designs and some of the different kind of flip flop designs. Now here we shall first be talking about some additional facilities that you can have in a flip flop.

And in particular how this edge triggering can be implemented then you shall be seeing how we can convert from one flip flop type to another in a general science fine.

(Refer Slide Time: 01:05)



Flip-flops with Asynchronous Preset and Clear

- It is often necessary to initialize a flip-flop to a known state before the circuit operation starts.
  - *Preset* (sets *Q = 1*) and *Clear* (sets *Q = 0*).
  - These are asynchronous operations, and can be easily implemented by using additional inputs on the cross-coupled gates.
    - The term "*asynchronous*" means that it does not depend on the clock.

~~0~~ 1 = $\overline{Preset}$
0 = $\overline{clear}$
$\overline{S}$
$\overline{R}$
1
1
$Q = ~~1~~\ 0$
$\overline{Q} = ~~0~~\ 1$
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

The first thing we look at is to see how we can implement a flip flop with asynchronous preset and clear what do we mean by this? See we have seen so many different types of flip flops SR, D, T, JK four types.

Now, we have seen how by applying the various input combinations, we can set the output to 0 set the output to 1 and so on for a T flip flop we can toggle it. But sometimes it may so happen there is a requirement that before you start this circuit operation you should reset or initialize the flip flop to either 0 or 1 and that need not happen along with the clock. It can be a so called asynchronous operation. Asynchronous operation means it does not depend on the clock; you just apply some input to output will be reset to 0 or 1. So, here we are talking about design of a flip flop with asynchronous reset and clear features. So, the motivation already I have mentioned that sometimes we need to initialize the flip flop to a known stage it may be 0 or it may be 1.

So, we define a preset operation which means setting the output to 1 and a clear operation which means setting the output to 0 and we shall see how these can be implemented very easily you can do it, and these are typically asynchronous operations which means they does not depend on the clock. Whenever you set preset and clear they will be activated immediately now let us see how it works. Well we look at one of the designs you are seen earlier namely that cross coupled NAND gates for designing a latch or a flip flop stage. So, we talk about that let us look at that cross coupled thing.

So, what do say you see for an SR latch for example, what we are said? We said that we are applying S bar here we are applying R bar here. Now here we are saying that let us have 2 additional inputs one apply to here another input apply to here. So, we are having 2 additional inputs to these two NAND gates which are directly generating the outputs.

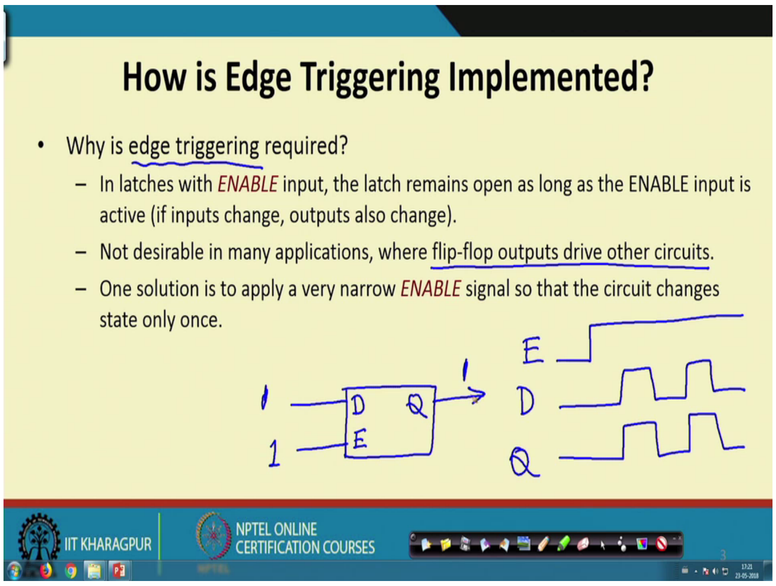
So, we are calling them let us call this input as preset bar, let us call this input as clear bar. Bar indicates that they are active low; that means, if they are set to 0, then you are activating that feature if it is 1 means you are not activating. So, suppose you are setting preset equal to 0, which means you are trying to present this latch.

Well if we apply 0 here you see what will happen these are NAND gate whenever you are applying 0, there can be clocks before because you see clock circuit is here I am not showing the clock circuit, this is the last stage of the latch I am showing. If you are setting a 0 then this Q will immediately become 1 and because it is 1 with this feedback this Q bar will become 0. So, we are setting or presetting the flip flop or this latch to 1 and similarly if you set clear to 0 then this Q bar output will become 1 and this 1 will be fed back and this preset is not active it is 1 of course, 1 1 and this equals to 1 this will become 0.

So, this is how you are presetting this to 0 or 1 by applying this preset and clear asynchronous input. Now this clock circuitry I am not showing which I generating this S bar and R bar in synchronism with the clock. So, when there is no clock, both S bar and

R bar are set to 1 and 1. So, under this condition whenever you are applying 0 to either preset and clear, the output will be asynchronously initialized to either 0 or 1 this is what this asynchronous preset and clear means.

(Refer Slide Time: 06:22)



How is Edge Triggering Implemented?

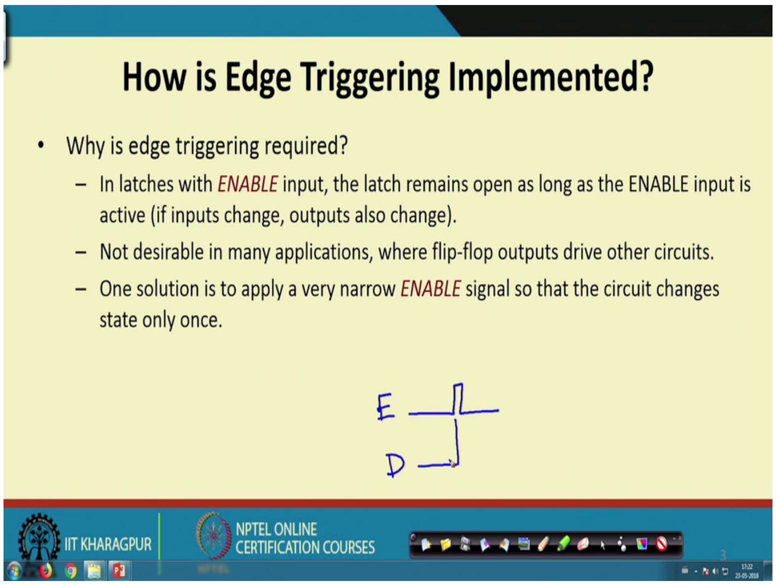
- Why is edge triggering required?
  - In latches with *ENABLE* input, the latch remains open as long as the ENABLE input is active (if inputs change, outputs also change).
  - Not desirable in many applications, where flip-flop outputs drive other circuits.
  - One solution is to apply a very narrow *ENABLE* signal so that the circuit changes state only once.

Now, let us look at how we can implement edge triggering, but before that let us see why we need edge triggering. So, why we talk about edge triggered operations well you think of a latch, let us say I have a D latch, think of a D latch where there is an enable input there is an enable input and there is an output. Suppose I have set enable to 1 and I keep it 1 for a long time what will happen is that this latch will remain one will remain open for the entire duration this enable is 1.

Like for example, if enable I make it 1 and I leave it 1 like this, and D suppose I change it make 1 0 1 0 in the mean time, what will happen this Q, Q will also go on changing as long as this enable is active; this is the problem that when you are enabling a latch for a longer time. So, any changes in the input will also cause a corresponding changes in the output unlike an edge triggered operation, where whenever an edge is there only then the output is changing all other times even if the input change output will not change ok.

Now, this is required for some application let us say where the flip flop outputs are driving some other circuit, this output you are connecting as input to some other circuit. So, if the output go on changing that output circuit can also start doing some wrong operations doing some wrong calculations that you may want to avoid. So, there can be several solutions the first solution we will look at is to somehow generate a very narrow enabled signal like this same flip.

(Refer Slide Time: 08:43)

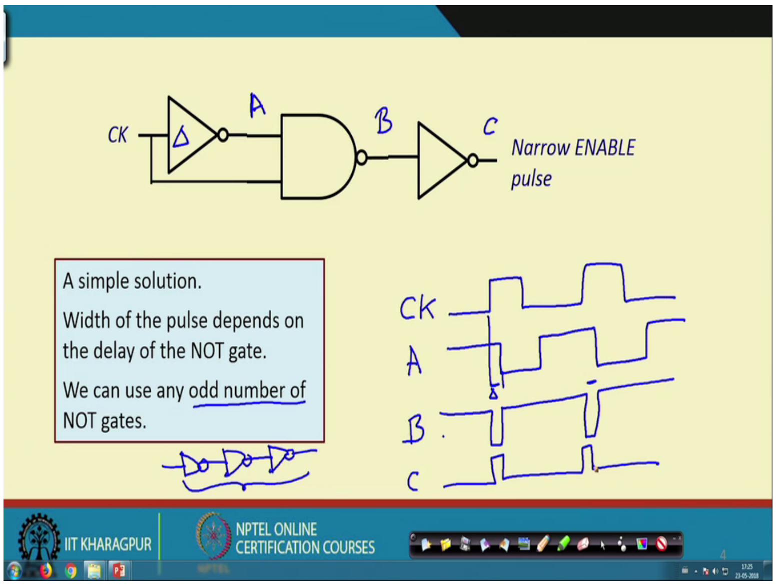


How is Edge Triggering Implemented?

- Why is edge triggering required?
  - In latches with *ENABLE* input, the latch remains open as long as the ENABLE input is active (if inputs change, outputs also change).
  - Not desirable in many applications, where flip-flop outputs drive other circuits.
  - One solution is to apply a very narrow *ENABLE* signal so that the circuit changes state only once.

What I am saying the enable signal will be a very narrow signal, very thin signal. And exactly at this point whatever is the value of D that value will be captured right. This is one solution, but now the question is how we can do this. Well doing this is not difficult well a simple solution can be like this.

(Refer Slide Time: 09:04)



A simple solution.

Width of the pulse depends on the delay of the NOT gate.

We can use any odd number of NOT gates.

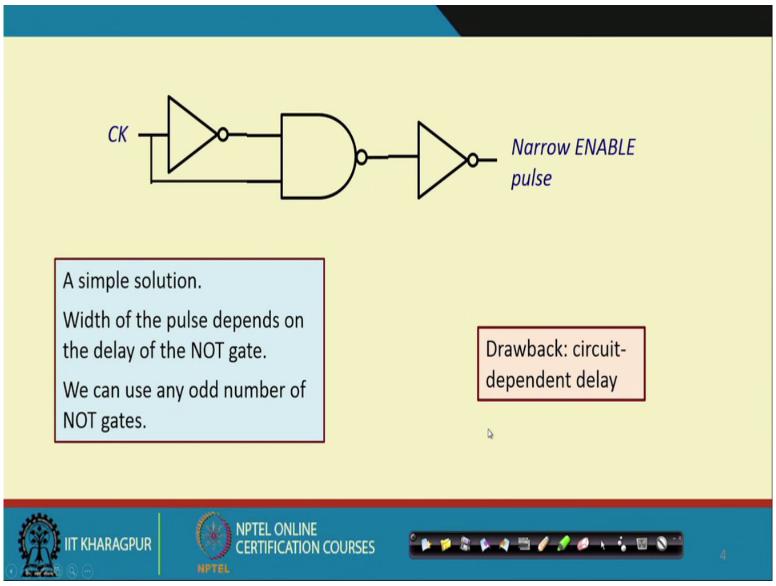
Let us try to understand, this circuit here we are saying that there is some kind of a clock signal we are applying let us say this is my clock signal. So, I am showing 2 such pulses let us call this as A, lets this call this as B and let us call this as C. Now A you see there will be a small delay of this NOT gate. So, if I plot A, A will be the NOT of this, but there will be a small delay after some small delay there will be a NOT. So, whatever clock you are applying. So, A will be just NOT of that, but there will be a delays small delay equal to the delay of this.

Now, you are doing an NAND of CK and A and generate B. NAND means what? So, whenever both of the inputs are 1 1 then only the output will be 0 otherwise the output will be 1. Now see here there is a period when both of them are 1 1 it is this period this period then again this period right both clock and a are 1 and 1.

So, you see 1 of a 0. So, B will be generating a pulse like this, now what will be the width of this pulse narrow pulse? The width of this pulse will be equal to the delay of this gate delta because this is nothing, but delta this edge and this edge the difference between the two, and because it is becoming negative just I am using another NOT gate to make it positive. So, I am applying a very narrow some kind of enable signal, which I can generating from the clock signal.

Now because it is so narrow. So, it will allow only one operation to carry one change in the output to happen. And this will avoid that earlier problem when enable is set to 1 for a longer time if the input changes output goes on changing continuously, that will also get avoided here. But well here another thing is that instead of one NOT gate we can use any odd number of NOT gates instead of 1 we can use 3 not gates also let us say. So, in this case this delta will be equal to the delay of the 3 not gates right, but the wave form will be very similar.

(Refer Slide Time: 12:19)



A simple solution.

Width of the pulse depends on the delay of the NOT gate.

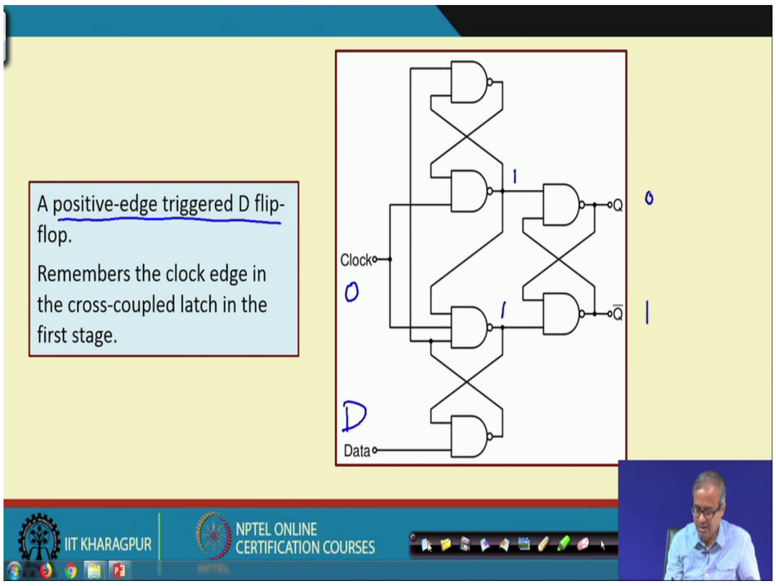
We can use any odd number of NOT gates.

Drawback: circuit-dependent delay

Now, we will one problem here is that, here we are getting a very narrow enable pulse all right, but the width of this pulse is not under our control, it depends on the delay of a gate which is circuit dependent.

So, if the delay if the width becomes too narrow, then the circuit may not respond to that very narrow pulse may be we have to make it little wider, but it is difficult to predict beforehand. Because when we are fabricating a circuit you are not very sure, what the exact delay of a gate will be there is always a plus minus tolerance during the fabrication process ok.

(Refer Slide Time: 13:01)

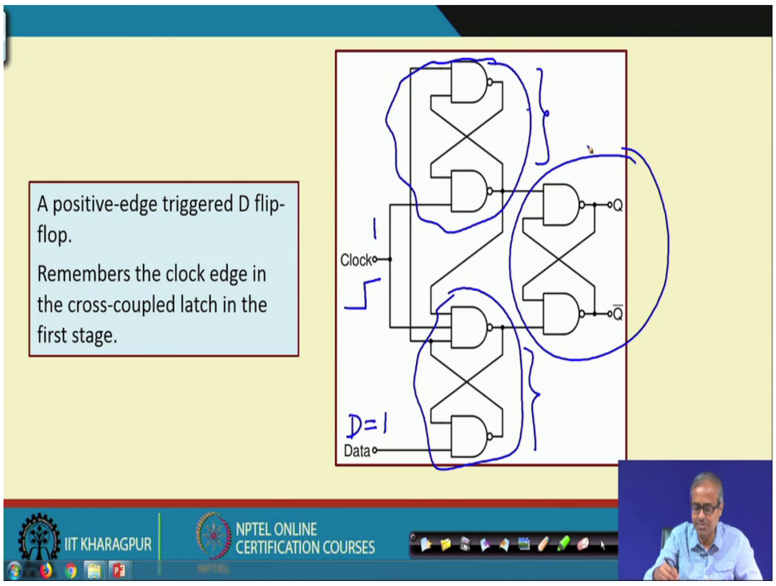

A positive-edge triggered D flip-flop.

Remembers the clock edge in the cross-coupled latch in the first stage.

So, better solution to have an edge triggered flip flop or a circuit is to have a mechanism like this.

Here we are showing the complete diagram of a positive edge triggered D flip flop where this is your D input and this is your clock input; let us try to see what is happening. Suppose when suppose when clock is 0, you see you see there is a clocks coupled latch on top there is one in the output stage and another here there are 3 cross coupled latch latches NAND gates. If it is 0 then this output will be 1 this output will be 1 which will make this work as a latch, there will be no change whatever Q and Q bar is there that same value will get stored if it is 0 and 1 it will remain 0 and 1, 0 1 means 1, 1 and 1 is 0 there will be no change.

(Refer Slide Time: 14:18)


A positive-edge triggered D flip-flop.

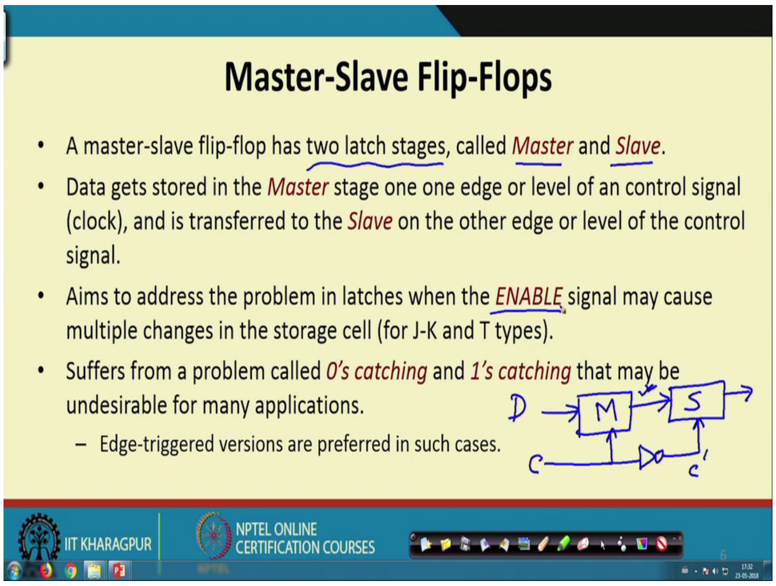
Remembers the clock edge in the cross-coupled latch in the first stage.

Now, let us see when the clock becomes 1 just from 0 it has changed to 1, it has just become 1 let us see what happens. And suppose at that time I have applied D equal to also 1. So, when this happens you see this D is equal to 1 is coming here right and this clock has become 1. So, actually what will happen this cross coupled latch, it will actually store this value 1 whatever you are storing here, this value will be triggering this here and this edge information will get stored here and it will not change subsequently. And in this one the value of D which was there at that point in time also gets stored and depending on that the operation of the flip flop will happened here.

So, without going into the detailed signals an all the lines what I am saying is that, they are 3 cross coupled latch stages we are using here, in one of the stage we are capturing the edge, if the edge that latch set to one immediately.

There is another D type stage where whenever D value is there and that clock is 1 that value will get stored there and the third stage is the actual D flip flop where the value gets transferred whenever the clock is active. So, this is the edge detection circuit and this is my actual D flip flop or D latch. As you can see this circuit is a bit complicated, but this works well this circuit remembers the clock edge actually there actually whenever it comes and it does not depend on the delay of the individual gates like in the previous approach. Well I am not going into the detail description of this circuit but I am telling you just this is a much better and robust design, and this is used in most of the implementation of edge triggered latches flip flops.

(Refer Slide Time: 16:44)



Master-Slave Flip-Flops

- A master-slave flip-flop has two latch stages, called *Master* and *Slave*.
- Data gets stored in the *Master* stage one one edge or level of an control signal (clock), and is transferred to the *Slave* on the other edge or level of the control signal.
- Aims to address the problem in latches when the *ENABLE* signal may cause multiple changes in the storage cell (for J-K and T types).
- Suffers from a problem called *0's catching* and *1's catching* that may be undesirable for many applications.
  - Edge-triggered versions are preferred in such cases.

Now, there is another kind of a flip flop which is used, I just as opposed to edge triggered this is called master slave flip flop. Now as the name implies there is something called a master, there is something called a slave and there are 2 latch stages, it is something like this there is one latch which you call as the master latch, there is another latch which you call as the slave latch.

So, the inputs you are applying to the master latch, the output of the master is going to the input of the slave and the final output of the slave is your circuit output. And the clock or the enable what you are here just applying you are enabling the master, and same signal after a not you are enabling the slave, this means that you are not enabling both the master and slave at the same time.

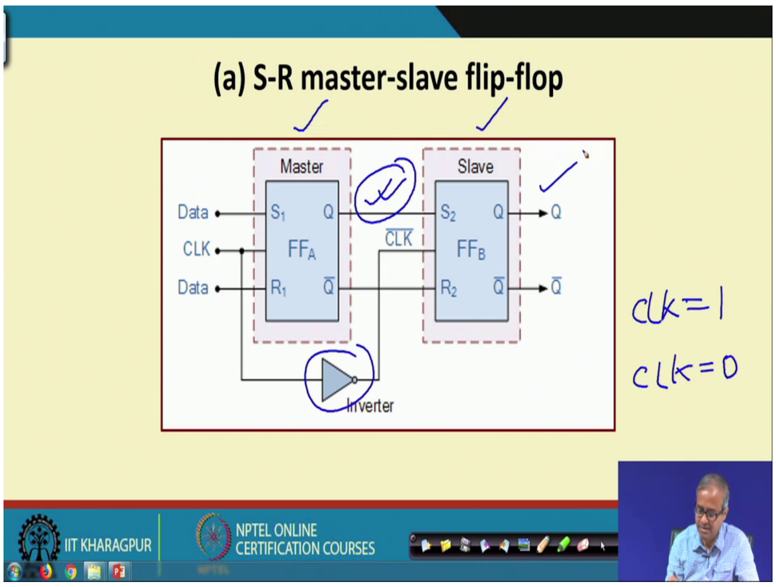
Let us say if it is a D flip flop you are applying a D. So, whenever this clock this signal control signal is active, the master is first active. So, the value of D will be transferred to the output of the master, but when this C is become deactivate this c prime will become active now slave will become active. This output of the master will now get transferred to this slave. So, you see if D changes multiple times in this process, there will be there is no harm because whatever is the value in the output of the master it is that value that will finally, go to the output of the slave and it will be a clean transition, there will be no multiple transitions like in a single latch that was possible right.

So, as I mentioned that this aims to address the problem in latches as I said earlier, when the enable signal may be active for a long time right and the output may change for multiple number of times. But there is some problem which may happen for master slave flip flop as certain cases which of course, here you are not discussing, this is something called 0 catching and 1 catching problem that for certain input combination and for certain means applications, the output of the master might be reset to 0 or might be reset to 1 depending on certain conditions.

But for many applications it is not an issue, you can use a master slave flip flop without any problem. But for those application where this is an issue then you have to use edge triggered versions and not master slave flip flops.

So, the pointer note is that master slave and edge triggered are 2 alternatives you are not using both of them together, you are using either edge triggered flip flops or master slave kind of flip flops both are used actually in practice.

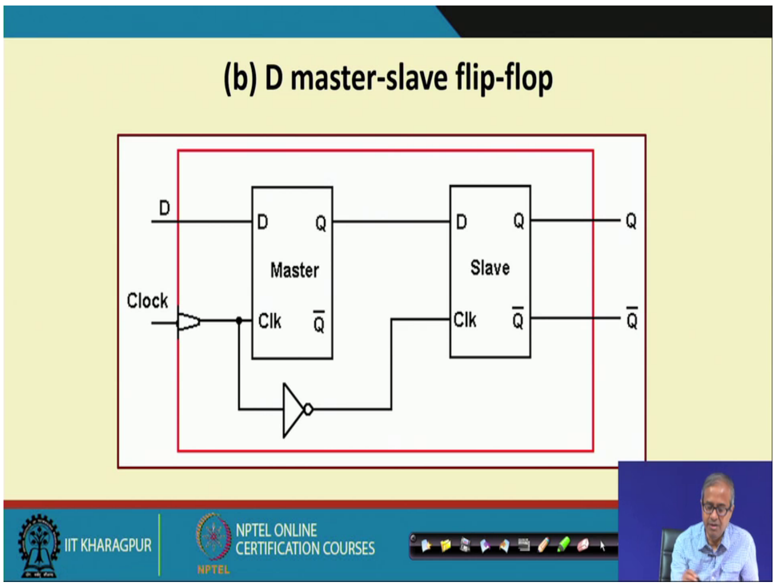
(Refer Slide Time: 20:03)



So, I am showing some examples of master slave flip flop designs; this is an SR master slave flip flop. See you see there are 2 stages master and slave both are SR flip flops and this clock or enable as I said you are applying it directly to the master and after an inversion you are applying it to this slave that same architecture.

So, when clock is 1 when clock is 1 master is enabled and whatever you are applying in the input accordingly the output Q is set and when clock becomes 0 then after inversion this slave will be enabled and whatever was here that will get transferred here right this is how it works.

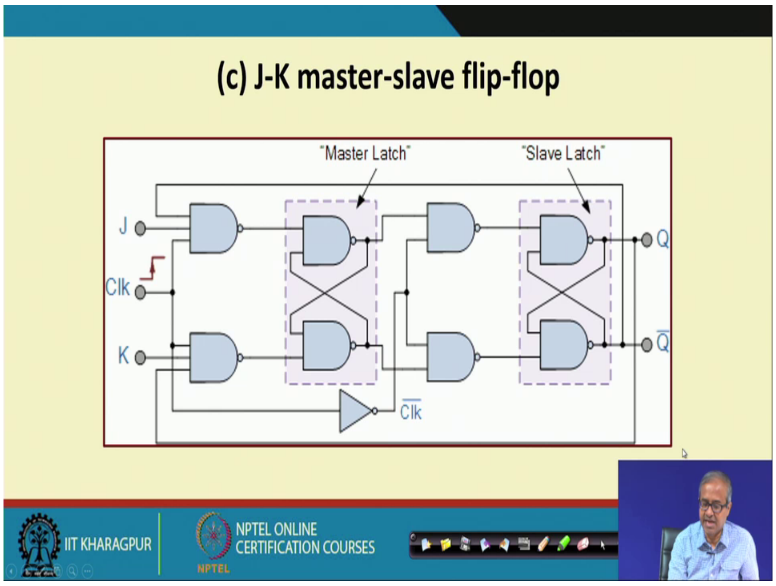
(Refer Slide Time: 21:00)



(b) D master-slave flip-flop

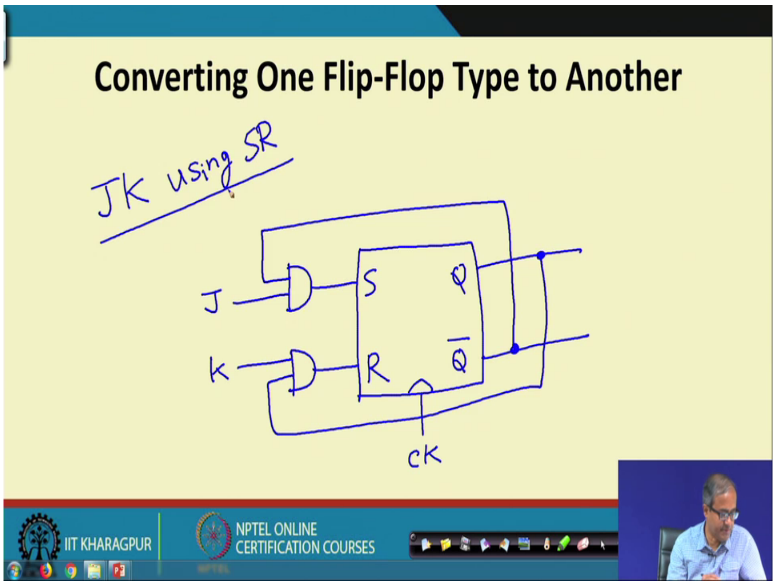
D similar, D already I have shown earlier, again in master and slave D is coming here clock and clock bar same.

(Refer Slide Time: 21:13)


(c) J-K master-slave flip-flop

And this is JK JK is also similar there is one stage here another stage master and slave, but here the only point is you note is that the feedback is coming from the final output of the slave back to the first gate ok. From Q bar to this J input and Q to this is k input this is your master latch, this is your slave latch this is master this is slave, the way the work are very similar.

(Refer Slide Time: 21:55)

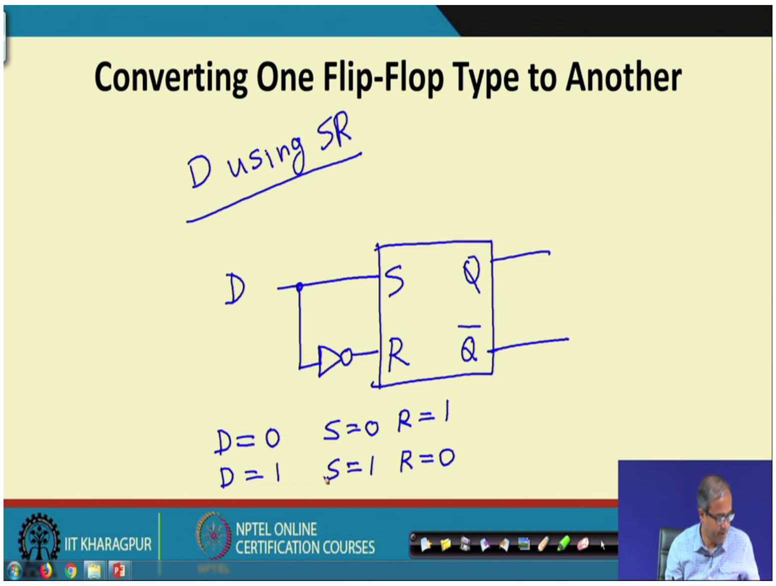

Converting One Flip-Flop Type to Another

Now, let us look into some ways of converting 1 flip flop types to another, let us look at a number of illustrative examples and see that how we can convert 1 flip flop type to another. Because there may be many designs where you are given 1 particular type of flip flop, let us say you are given JK flip flop, but you actually required a D type flip flop. So, how to convert it into a D type flip flop that you should know ok. So, let us see this through some examples one by one. The first example that we work is let us say just one by one let us see sorry yeah. First we see we look at JK flip flop using SR flip flops.

So, we are given SR flip flop. So, how to build a JK flip flop let us see. So, you have an SR flip flop given with you, this is given and you have to build a JK flip flop of course, clock is there clock I am showing here let us say clock is here. So, what you do? You use 2 AND gates and we apply the J and K inputs here J and K and what do you from Q bar, you take a feedback to here and from Q you take a feedback to here. So, you get a JK flip flop. This is how you actually built a JK flip flop using SR flip flop. You see to built a JK flip flop the essential idea is same. You have to take a feedback from Q bar you have to bring it to the gate that feeds J and from Q you have to bring it to the gate that feeds K.

So, I suggest you can work out this condition for the different input combination C that this actually works as a JK flip flop I will leave it as an exercise for you. So, this was JK using SR.

(Refer Slide Time: 24:43)



Converting One Flip-Flop Type to Another

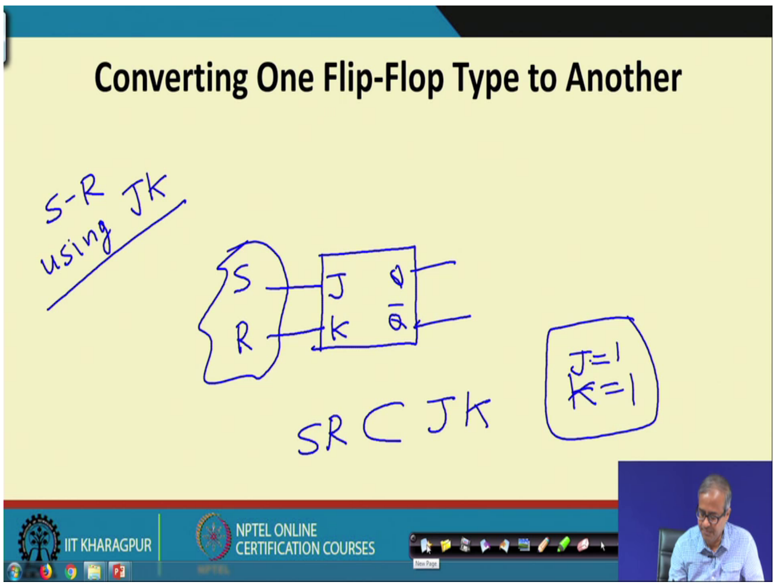
D using SR

$D=0 \quad S=0 \quad R=1$

$D=1 \quad S=1 \quad R=0$

Let us next look at a simpler problem suppose we want to build a D flip flop using SR flip flop, D using SR. Let us say again similar suppose we have an SR flip flop again Q, Q bar, this is very simple you connect D directly to S and with an inverter NOT gate you connect it to R. The idea is when you are trying to apply D equal to 0; that means, you are trying to store that 0 then you make S equal to 0 R equal to 1, that will make Q equal to 0 and when is D equal to 1, you want to store that 1 you make S equal to 1 and R equal to 0 that will make it 1 output equal to 1 fine.

(Refer Slide Time: 25:58)

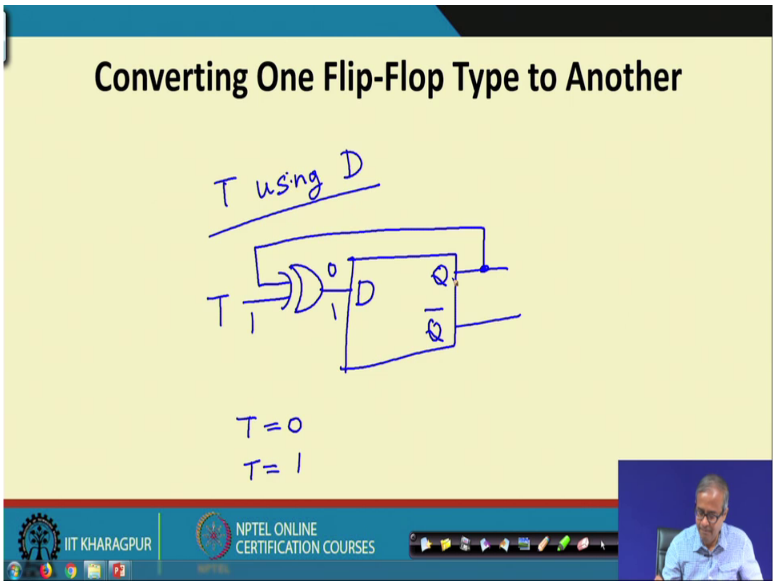


Converting One Flip-Flop Type to Another

Next let us look at S R using JK. You see here you do not have to do anything sorry if you have a JK flip flop with you, if it is J K you simply call them S and R Q, Q bar are there you do not do anything why? Because I mentioned this SR flip flop is functionally it is a proper subset of JK flip flop. It uses only some of the functionalities of JK see J equal to 1 K equal to 1 that combination it is not using.

But it does not matter because you will never apply S equal to 1 R equal to 1 because you are using an SR flip flop. So, it is a JK flip flop where you are never applying this input combination. The remaining combination are identical for an SR flip flop. So, it does not matter. So, here you do not have to do any kind of change it is just straight forward.

(Refer Slide Time: 27:17)



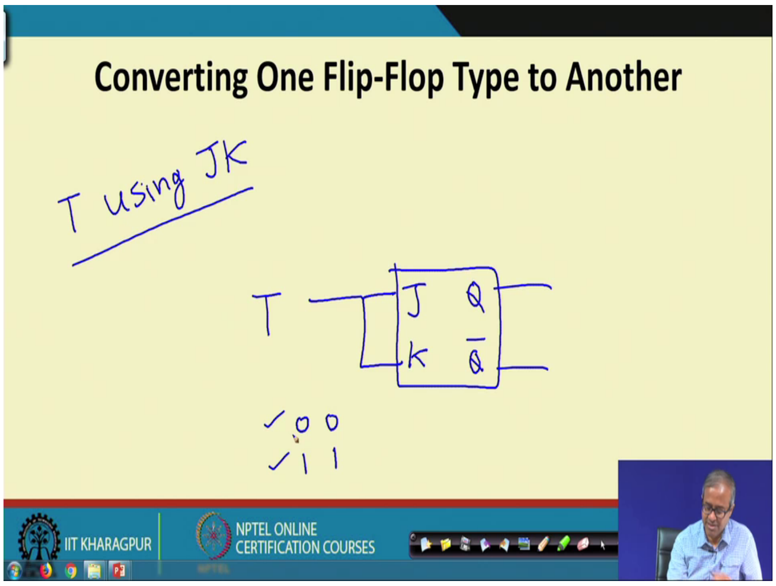
Converting One Flip-Flop Type to Another

Let us look into another one, T flip flop let us see using D flip flop see in the T flip flop you have talking about some kind of toggling. So, you have a D flip flop with you D Q and Q bar.

Here you use an exclusive OR gate you use an x OR gate you apply the T input here and this Q output this is fed here just see what this means. If T is 0 then there is not suppose to be any change, see if T is 0 then the same Q same value will be coming here, that same value will be gets stored is a D flip flop if it is 0, 0 will be coming here if it is 1, 1 will be coming here. So, no change and if T equal to 1 there is supposed to be a toggle, if Q equal to T is 1.

If Q is 0 then this will become 1 0 x or 1 and if Q is 1 it will become 0 1 x or 1 0 there is a not toggle and that toggled value will get stored ok. So, it is a T flip flop right.

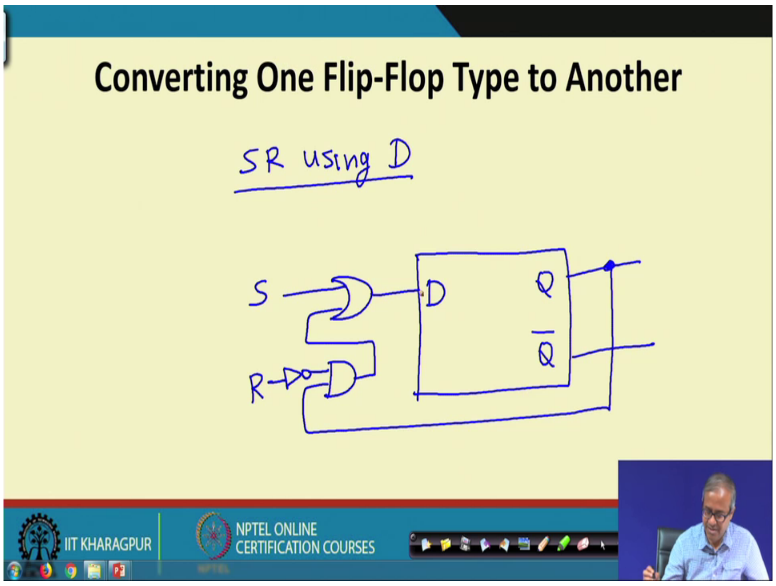
(Refer Slide Time: 28:55)



Converting One Flip-Flop Type to Another

So, let us look at a some others a T using JK I already mentioned earlier let us let talk about it once more T using JK flip flop. So, if we have a JK flip flop, you simply tie J and K together and call it T that is done. Because again for a T flip flop you are using only one of the rows of JK flip flop not 1, actually 2 when the combinations are 0 0 J 0 K 0 and J 1 K 1 ok. So, when T equal to 0 you are selecting this T equal to 1 you are selecting this just that.

(Refer Slide Time: 29:47)

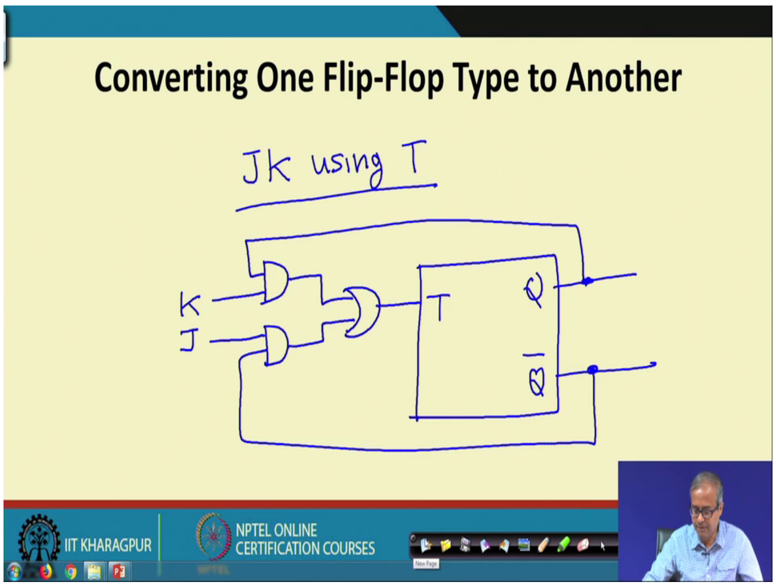


Converting One Flip-Flop Type to Another

Then let us look into a slightly complex one. Suppose you are having a D flip flop and you want to build an SR. Here you have a D flip flop given to you and you use an OR gate which you connect to D this you connect to S suppose I have AND gate, the output you connect here R with a knot connect here and the Q bring it here. So, I leave it as excess for you to check that this actually works as an SR flip flop. Because you will see if S equals to 0 R equal to 1 this will make D equal to 0; that means, you are storing 0 if S is 1 R equal to 0 it will make D equal to 1; that means, you are storing 1.

But if it is 0 0 then the previous value is coming and that Q value is getting stored here that same value is getting stored right.

(Refer Slide Time: 31:27)



Converting One Flip-Flop Type to Another

And the 1 last kind I am showing JK using T. Suppose you have a T type flip flop you want to build a JK flip flop. So, you use an OR gate here connect it here connect like this, you apply the K input here, apply the J input here and the Q input you connect here to K and Q bar input you connect it to J this makes a JK flip flop. So, here again I will leave it as exercise for all of these 2 for you to verify that whether this actually performs as the JK flip flop by applying all possible input combinations and verifying whether this state table is actually being satisfied fine.

So, with this we come to the end of this lecture, where we talked about various features in a flip flop like edge triggering how edge triggering can be implemented, and lastly we talked about how you can convert the various kind of flip flops from 1 type to the other.

Now in the next couple of lectures we shall be discussing some of the other clocking and timing issues in sequential circuits, which will be very useful to understand whenever you are designing sequential circuits which will be taking up in the later discussions. These timing issues will be very important for you to understand and appreciate some of the concepts that we shall be discussing later.

Thank you.