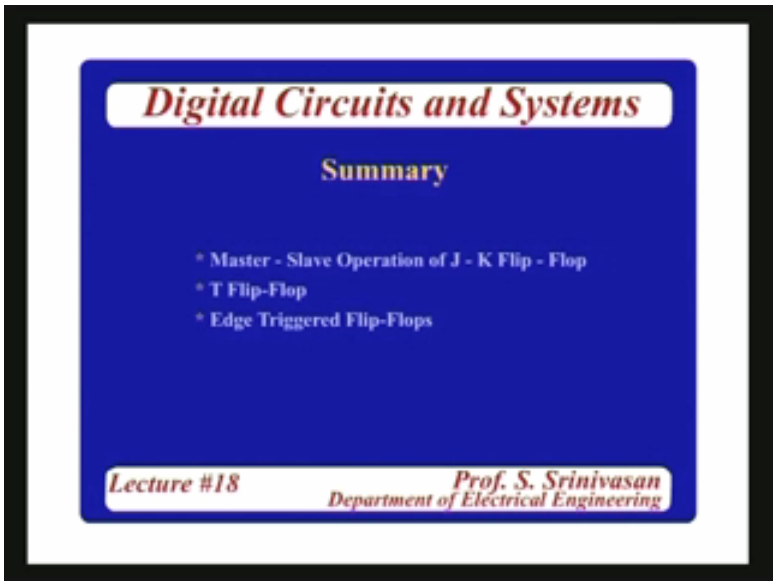
Digital Circuits and Systems
Prof. S. Srinivasan
Department of Electrical Engineering
Indian Institute of Technology, Madras
J-K and T Flip Flops
Lecture # 18

(Refer Slide Time: 2:05)



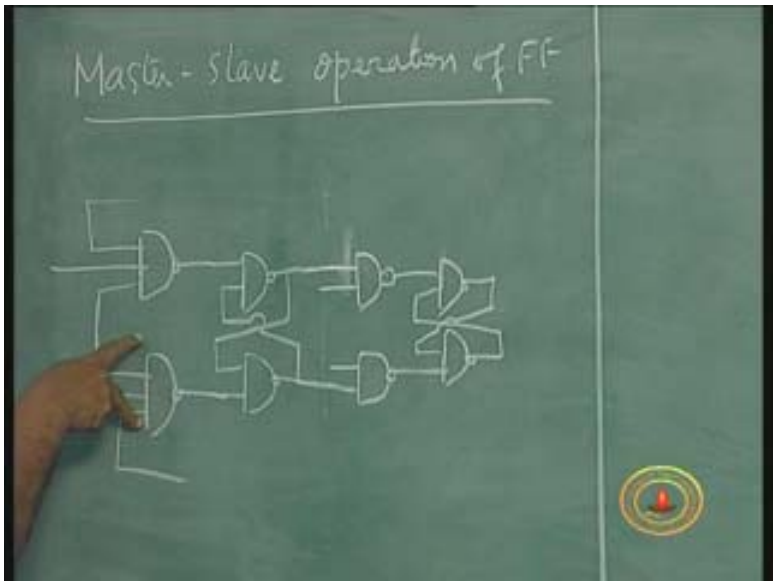
In the last lecture we have been seeing about the flip-flops as against the latch. In the latch you store a bit 0 or 1, in flip-flop also you do that but a flip-flop has an additional control called clock. When the clock is high only then the data can be stored in the flip-flop and when the clock is low whatever has been stored will be retained even when there is a change in the input.

We saw the basic SR flip-flop and the D flip-flop wherein a single bit data can be stored with single input D data input by connecting S and R through an inverter. Then we talked about the condition of 1 1 being not permitted for an SR flip-flop and see whether we can do something about it. So we modified the input AND gate into a three input NAND gate and found that it is possible to have an operation corresponding to S is equal to 1 R is equal to 1 wherein the flip-flops output keeps changing constantly from 0 to 1 to 1 to 0 and 0 to 1 to 1 to 0 so forth as long as the clock is high. When the clock is low of course it retains its content as a memory state.

Now the very frequent change of the output with an SR flip-flop with S is equal to 1 R is equal to 1 with an input NAND gate is disturbing because we don't want that type of behavior, it is called racing I said, racing is not a desirable behavior. On the other hand it is a nice idea to have an input condition with 1 1 where the output changes from the present state to the complementary state. If the state had the output of 1 you will like to

have a 0 output and from 0 to 1 back and forth 1 and 0 0 to 1 provided it does only once in a clock period. So we were thinking about how to prevent this racing and one way to do it is to make a clock active period much much smaller, very small, smaller than the propagation delay of the flip-flop. This is not a practical solution. There are couple of other ways we can do it. One way is called the master slave concept.

(Refer Slide Time 6:50)



So what we will basically have in a master slave operation?

The input gates remember this and this went to the two NAND gate combination (Refer Slide Time: 5:48) this is the basic structure we discussed in the last lecture and this came from here and this came from here. Instead of that what I am going to do is to extend this into one more stage of the same identical flip-flop and the output of this I will feed into the SR latch as we did earlier. This is corresponding to the clock input. Now instead of tying the output of these two input gates what I will now do is to take this all the way from here after this second stage (Refer Slide Time: 6:58) I am duplicating this into this except that I do not need three inputs I need only two input because I am going to give external input only once. These are the external inputs, this is the input from the clock, this is the feedback now output of this will be fed into this, this is the latch with part of this flip-flop.

Now I will have an identical stage of flip-flops where the output of this will be fed into this, output of this will be fed into this and then I have this basic SR latch here. This output will go into this (Refer Slide Time: 7:46), I am taking the feedback all right from the output but not immediately but after one more stage, one more similar stage I put and then take the feedback. So this I will call Q and Q bar, these are intermediate outputs and this also will be clock but one thing I will do is I will not connect the clock as it is into this but I am going to invert my clock and then connect this.

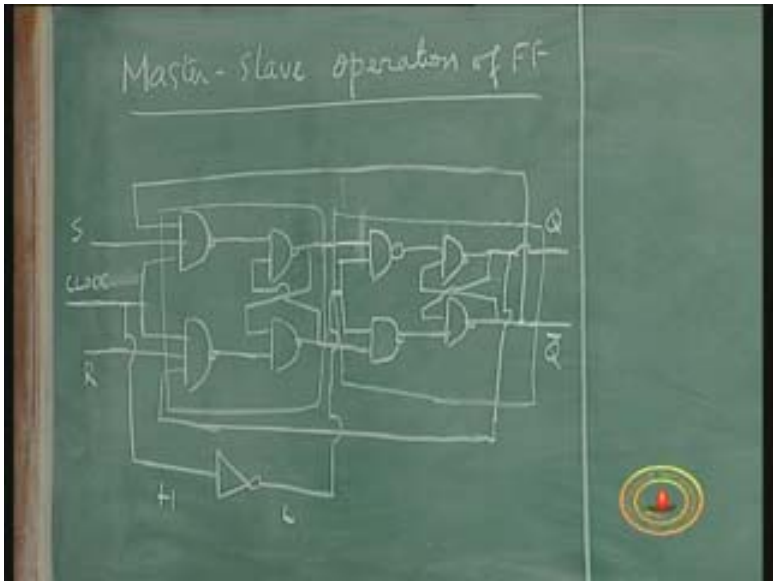
So now I have more or less a condition of two flip-flops, the clock is common but one of them is fed directly and the other clock is fed through an inverter, these are the two external inputs which we call S and R earlier and the third input comes from the output stage not immediately but after one extra stage the second stage. You put a second stage and take the output of the second stage and feed it back into the input.

Now this will behave exactly like the feedback flip-flop like we discussed yesterday except that this output will change only once in a clock cycle I will tell you why. Now the clock has to be high for any normal operation the flip-flop that's what we said. The function of the clock is to enable the output to change as per the inputs when the clock is high and not change the output when the clock is low whatever the input is. Now, if that is the case I am feeding a clock here but an invert of the clock here. The clock is high for this flip-flop and it is low for this flip-flop.

So if we consider this as one flip-flop whose output is fed as input to the second flip-flop, I will call this second flip-flop (Refer Slide Time: 9:55) this flip-flop and this flip-flop are identical of course there is an extra input is there the feedback input so this gate is a two input gate and this gate is a three input gate that is one difference but that is because I want the feedback only once. But more importantly the difference between these two flip-flops is that one has the clock which is high and the other is low and vice versa.

Therefore, when the clock is high for this flip-flop this flip-flop will change state but this feedback is not going to happen because feedback is from here and not from here so this is constantly changing and the racing condition is not going to happen. Only when the output is fed back into these two gates because of the property of these NAND gates the input keeps constantly changing and the output keeps constantly changing. This flip-flop has a clock low when this clock is high. So when this clock is high this clock is low and when this clock is low whatever is the input then, it is not going to affect the flip-flop because the output is going to stay the same in its previous state the memory state when the clock of a flip-flop is low.

(Refer Slide Time: 11:15)



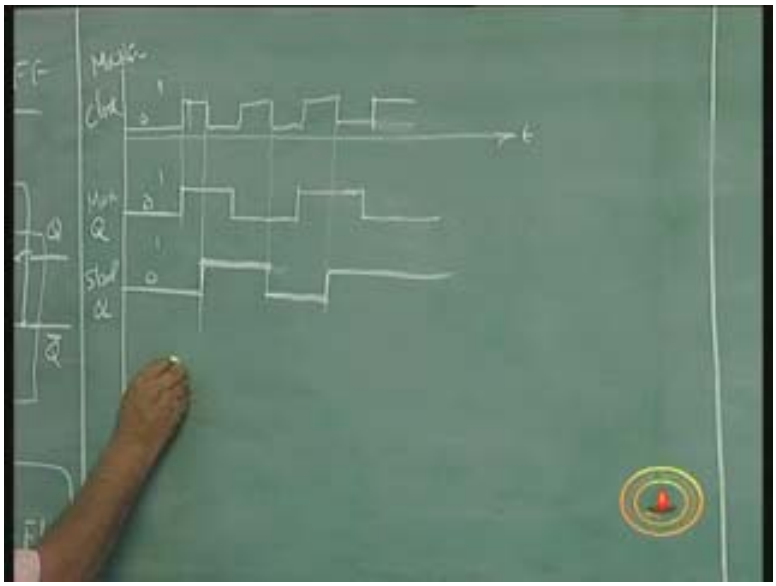
So when this flip-flop is enabled or the clock of this flip-flop is enabled the clock of this flip-flop is disabled. Any change that is going to occur is going to occur here, anytime it can happen but only when the clock has completely changed from high to low here which means it is low to high here. Whatever is the output here will now go into this so the output of this flip-flop goes to the input of this flip-flop and it will change the output of this flip-flop when the original clock is low. But at that time the feedback in the changed signal cannot affect this gate and the output of this gate cannot change now because at this time the clock is low again. So the output fed back into the input cannot affect the operation of the circuit until the clock changes into 1 again. That means I am preventing this continuous variation, the variation can only be once, when the clock is high any change that may happen in this flip-flop can only happen locally and that local change will be transmitted into this flip-flop only when the clock of this flip-flop changes into high at a time this is closed. So any change that may happen because of this change at the output will be fed back but will not be recognized until the clock changes back to 1 again to original clock.

So I have a two stage flip-flop basically where the 1 stage clock being high the other stage clock being low at a given time and after some time the first stage clock becomes low and second stage clock becomes high so that way I am isolating the fluctuations from the output or output changes will not be reflected in the changes in the input stage. Such a combination is called a master slave combination, this flip-flop is called a master flip-flop and this flip-flop is called slave flip-flop (Refer Slide time: 13:36). The slave can only take the master's output and convert into its own output. But that can happen only when the clock is enabled for the slave, and if the clock slave is enabled then already the master clock is disabled.

So the output changes in the slave flip-flop will not affect the master output until the clock of that flip-flop becomes high which is only when at the end of that period, for this

clock it is at half period. that means we are now making sure that changing from Q to Q bar when S is equal to 1 R is equal to 1 may still happen but only once in the clock period, when the clock is high when this master clock is high.

(Refer Slide Time 19:00)



Instead of this toggling like we saw, this is the clock, this is the master clock as a function of time, it moves from 0 to 1 master output originally it would have been like this that is because of the feedback. But now the feedback is not there. So, if it was 0 earlier it will become 1 and then it continues to be 1 and until again it becomes 1 at that time it will become 0. So this change over from 0 to 1 to 1 to 0 will happen but only once during a clock period.

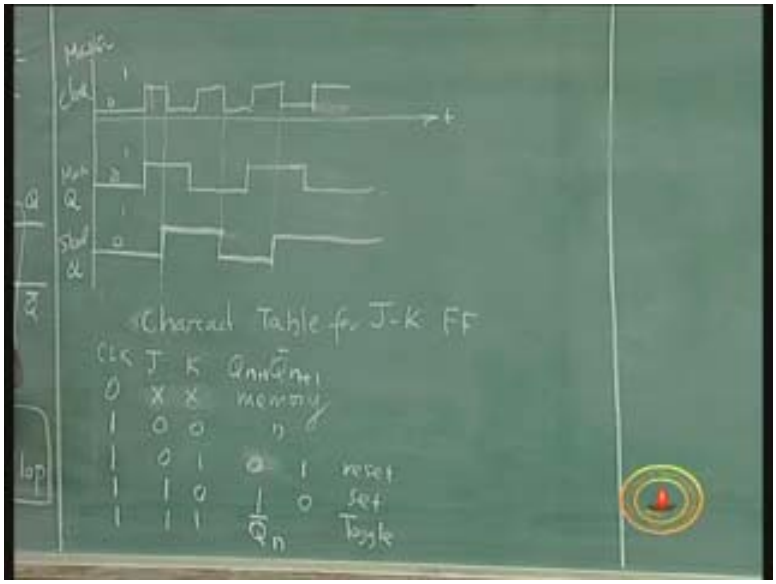
So the output of the slave will be only this, this change will be reflected into the output only at the end of this clock period (Refer Slide Time: 15:37) so that will be during the positive edge, at this point the slave clock will be low, when the master clock is high slave clock is low so the change of the master cannot go to the slave, at this point the slave clock becomes high so when the slave clock becomes high only then the master output would be recognized by the slave so the slave clock would be working like this. I mean slave output will be delayed by this clock pulse period. So the output keeps changing from 1 to 0 also here in this case just as in the case of the previous circuit like we discussed in the last lecture. Constant toggling is not there but it changes only once in a clock period. This is a toggling mode really. As I said toggling is a controlled change, racing is an uncontrolled change, toggling is useful we will see that later on how it's useful, racing is not useful, and racing is undesirable.

That means I have eliminated now S is equal to 1 R is equal to 1 problem which we had the original SR flip-flop. In the original SR flip-flop 1 1 condition was not to be used because as because for certain unreliability undefined performance, uncertain performance we want to remove that but we ran into a racing problem and we solved the

racing problem by putting an extra flip-flop so this is a master slave configuration, it can work for any type of flip-flop, for D flip-flop also you can have a master slave. SR I have now made it like this, you can have two flip-flops with one clock connected to the master clock and slave clock with face inversion such a configuration is called a master slave configuration. It doesn't have to be for only SR flip-flop. That means this new flip-flop is tamed for S is equal to 1 R is equal to 1 condition so I will now give a new nomenclature because the SR flip-flop will always bring back the memories of 1 1 being not permitted so I don't want to call it a modified SR flip-flop or anything so this SR flip-flop with the feedback and an extra stage to control the operation this whole configuration is called a JK flip-flop, I don't know why it is called JK flip-flop.

JK flip-flop works like this; so instead of calling this S input I will call it J input same as S input except for the case of S is equal to 1 R is equal to 1 SR flip-flop and JK flip-flop are identical. JK flip-flop works exactly like SR flip-flop for the three cases 0 0 0 1 1 0 for 1 1 case SR flip-flop is not permitted but JK flip-flop is permitted with the toggling output its all where you stay.

(Refer Slide Time: 26:05)



So SR flip-flop plus this additional toggling facility becomes a JK flip-flop so this is called J and this called K so J K Q Q bar and we will also put a clock here when the clock is 0 no matter what happens J and K, output is a memory state, when clock is 1 0 0 is memory state, 0 1 is the reset state that means 1 1 0 1 0 this is the reset this is set 1 1 1 so what is Q we don't know until we know what Q was Q toggle means that the previous value of Q is complementary the present Q is the complement of the previous Q so you can write toggling mode you write Q bar so you can't write Q is equal to Q bar it does not sort of looks odd so what I should write is at this clock pulse if it is nth clock pulse what is the value of Q is for n minus 1th clock pulse will become the value now. so I will call this n this will be the previous value $Q_{n\ plus\ 1}$ would be Q_n plus Q_n complementary, Q_n complementary becomes $Q_{n\ plus\ 1}$ or $Q_{n\ minus\ 1}$ complementary becomes Q_n so this is the

toggling mode, this is a sort of a flip-flop in which all the conditions are made, we do not have to worry about 0 1 1 being encountered in your circuit as inputs and toggling has its own use we will see it later, you design counters toggling property has used.

So now to summarize we have now seen an SR flip-flop of course it is sort of a latch we can store a bit of information and keep it there and tell you change it then we want to do it in a controlled way whenever we wanted it that means introduce a clock and say when the clock is high only the inputs should be recognized so it is a clocked SR flip-flop and then we said we want to put a simple data bit 1 or 0 and thought of why we need two inputs so we converted an SR flip-flop into D flip-flop so in the D flip-flop with the clock whatever data you put in gets stored and with clock 0 whatever you put in the data does not get in but it remains the same as what it was earlier. And we modified the SR flip-flop into an extra gate with the feedback extra input of the input gate with a feedback and we called it a JK flip-flop where for that condition of S is equal to 1 R is equal to 1 or in this case J is equal to 1 K is equal to 1 we had the output complementary to what was earlier in the previous clock cycle and in order to avoid racing we had to put two of this in a master slave mode or master slave configuration, so this is the JK master slave flip-flop they call it.

In the D flip-flop also you can have a master slave operation. If we have two D flip-flops as one connected to the clock high and the other connected to the clock low at the same time and when you put one value here that value gets stored immediately when the clock is high. But then that D cannot transmit into the next flip-flop because the clock of that flip-flop is low. As the clock becomes low for the master flip-flop so it cannot take a new value. Hence the value stored in the flip-flop will go into the second flip-flop the slave flip-flop because the clock of that will still be high. So I can have the data stored in the second flip-flop when the clock is low instead of having to store data when the clock is high. It can happen only once because the master is now 0 clock and whatever change I make in the D input, if you want to make only one change in your data per clock cycle then this is the best way to do it. If you want to make only one change in the data storage in one clock cycle or in one clock period you put a master slave configuration. The master takes it and the slave will not be affected. And whatever happens in the master will be transmitted to the slave at the end of the half period when the clock becomes low from high and at that time the slave takes the output of the master whatever be the value. But any more change in the master cannot happen because the clock of the master has already become low.

So to avoid racing this JK concept can also be applied to a D flip-flop a master slave D flip-flop is also possible. I said there are two ways of making sure that the data changes only once in a clock period. One is to make sure that data changes when the clock is low and when the clock is high do not put the data that is one way and the other way is to have a master slave configuration. Whenever the data changes the output also changes but for only once. The second stage data changes when the clock goes from high to low. It is the same case in a D flip-flop you put a D value the master will take it during the clock high but then it will pass it on to the slave the clock goes from high to low that is a

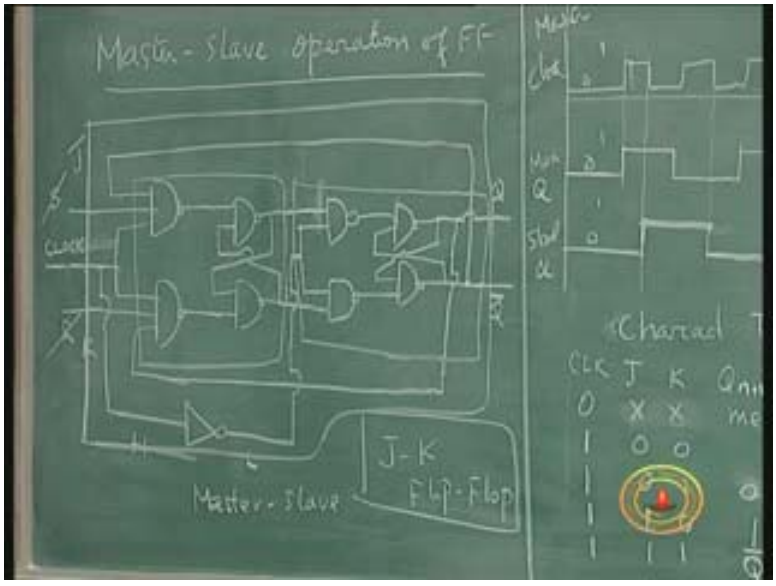
master slave D flip-flop. The master slave is a concept but it helps to avoid racing in the case of JK.

That means we have all the combinations. In D flip-flop it is just simply storing the data, in SR flip-flop it is sometimes but most probably SR flip-flop is no use because you want to simply store data you put a D flip-flop if you want more than that the toggling action you put a JK flip-flop. There is no need to have an SR flip-flop. Generally SR flip-flops are not used that much. They use D flip-flop just for data storage purposes keep it and store it and whenever you want get it back or if you want to do some other thing with a flip-flop like a toggling action which I said which will be used in counters then use a JK flip-flop. These are the two most popular flip-flops.

Sometimes I want only this toggling action. in that case what I should do is I will take a JK flip-flop, now whatever I draw here as JK is the whole thing, these two inputs and the clock is only available to me and output Q and Q bar at the master slave are only available, internal feedback, the internal inversion of the clock are those things that are not available to me in a block. When I buy a JK flip-flop I have a single hardware where there are two inputs for J and K and one input for clock, one output for Q, and output for Q bar, the Q and Q bar outputs are from slave and JK inputs are into this master and the internal clock gets inverted for the slave.

Of course you need to give some power supply no circuit will work without powering up, we need voltage and ground. You need two terminals; one to give the voltage called V_{cc} or V_{dd} depending on whether it is a Bipolar transistor technology or CMOS technology and then ground.

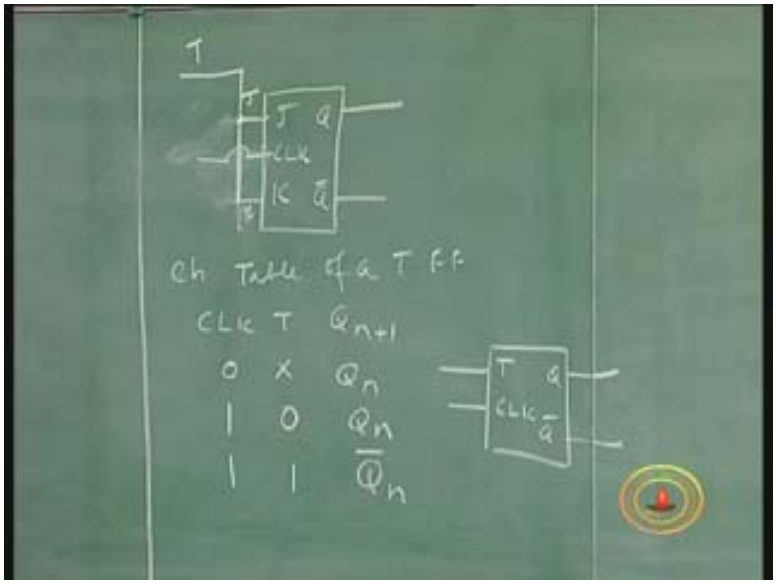
(Refer Slide Time: 28:29)



So I am not talking about power source how are you going to power up them? I am only talking about signals, JK or signals, output as a signal, clock is the signal. You have to

know the difference between the signals and the power source. Power source is required. For any active circuit we need a power source. We will not consider power source in this drawing. Usually we will have to put in addition to that V_{cc} and a ground and all that. So JK and clock and Q and Q bar. If I only want toggling action, I have an input, I have a clock period and I want an output which is always toggling back and forth at the data 1 per clock period then I don't have to go through this SR flip-flop type of thing I can as well connect these two together and J is equal to 1 and K is equal to 1 I will call this T input T for toggle, toggle input, when T is 1 it toggles and T is 0 it is the memory state. When T is 0 it is 0 0, J is equal to 0 K is equal to 0 is a memory state, J is equal to 1 K is equal to 1 is toggle state that is why the input is called toggle input, when T is 1 it toggles and T is 0 it doesn't toggle or it stays and of course clock is required.

(Refer Slide Time 31:40)



So what will be the truth table of this or characteristic table of a T flip-flop? Clock T $Q_{n\ plus\ 1}$. When clock is 0 output remains the same no toggling action, and if clock is 0 it is the memory state always. For all flip-flops if clock is 0 it is a memory state, when clock is 1 and T is 0 again it is memory state. When clock is 1 high if I don't give the toggle input which means it T is equal to 0 which means J is equal to 0 K is equal to 0 that means it is a memory state. When clock is 1 and T is 1 then this is Q_n bar.

The previous value of Q_n gets complemented of the new value, the next value of Q_n is always the previous value complemented. So this is the toggle flip-flop the T flip-flop, sometimes you get a T flip-flop these are the two inputs that means I have to assume that internally it is a JK flip-flop where they have tied T and J and K. In the D flip-flop I said internally S and R are inverted. If there is an inverter between S and R then it's called a D flip-flop, likewise this is called T flip-flop. That means there are totally four types of flip-flops we have seen; SR flip-flop with its own limitation of 1 1 condition, a D flip-flop where you can only put 0 data storage and not a toggling action and you put a JK flip-flop where it works for both the D flip-flop and a toggle flip-flop. So one JK flip-flop is equal

to 1 SR flip-flop or one D flip-flop plus 1 T flip-flop and a T flip-flop only toggling happens. That means we can use any of these four flip-flops for the particular operation or performance of a circuit that we design.

This extra hardware this slave hardware almost doubling the thing effort in terms of the number of gates hardware one of our goals what is one of our goals always been saying three things high speed, low power, low cost or small size so that means we are unnecessarily putting extra hardware. So, is there anyway where I can do the same changing only once in a clock period but not having to put an extra flip-flop duplicate an extra flip-flop? That is one reason why I may not used JK flip-flop sometimes. There is another reason why we may not used JK flip-flop. We said what is the fastest rate at which the clock can change? The fastest rate at which the clock can change is the time it takes for the propagation delay of the two flip-flops.

Any flip-flop is a single flip-flop. How fast can I change the data and keep storing it or sending it out? At the rate the propagation delay is the time it takes for the data to travel. Of course in addition to propagation delay there are couple of things called set up time and hold time we will see this things a little later on in the course right now don't worry about this terms there is a set up term, there is a hold term and the propagation delay. Let us bunch them all together for this discussion today and there is some delay involved. Within the delay I cannot change the input. That means if I clock it faster than that period corresponding to the delay then it is not fast enough to catch up. Any change in the input cannot catch up with the, the output cannot catch up with the changes in the input faster than the total delay of the flip-flop which includes propagation delay, setup time and hold time.

When that is the case of course in extreme cases only this is a problem. Most of the times in today's technology the speed is so high we don't have enough applications. There are very high speed applications where the speed is not really enough, that is the reason they go on improving the technology and inventing newer and newer circuits. That is because there is always a cutting edge technology cutting edge application. But in most of the applications today's technology is so good where we don't have to worry about all these things such as the propagation delay of the flip-flops. But still from a theoretical point of view you should know the limitations so the limit within the change of clock the limit at which the data can change is the period and if I try to change the clock faster than that then data cannot catch up with that.

So if we now put two of them then there is going to be a slow down further. If the propagation of the flip-flop or the delay of the flip-flop that involves several things including the propagation delay then that is the limiting factor in my operation. There are conditions, applications, places where the delay of the flip-flop is the limiting speed it is not going to help me. If I had to put two of them it is going to reduce by factor two so I have to slow it down by factor of two and I am already worried about my cutting edge application where the speed is not enough. Now if I have a master slave configuration I need to have to do it at half the speed.

There are always some extra things coming in here and there but approximately it is half this speed at which I can work a single flip-flop because it is two flip-flops two delays. That is one reason why we want to think of another scheme at the same time not allowing the racing, this is important. Racing is a very bad thing because you know want to have unpredictable performance. So without racing and without master slave is there anyway we can do it so that only once in a period once in a clock period the output can change. The second thing there is extra hardware I need. So there is a technique called edge triggered technique.

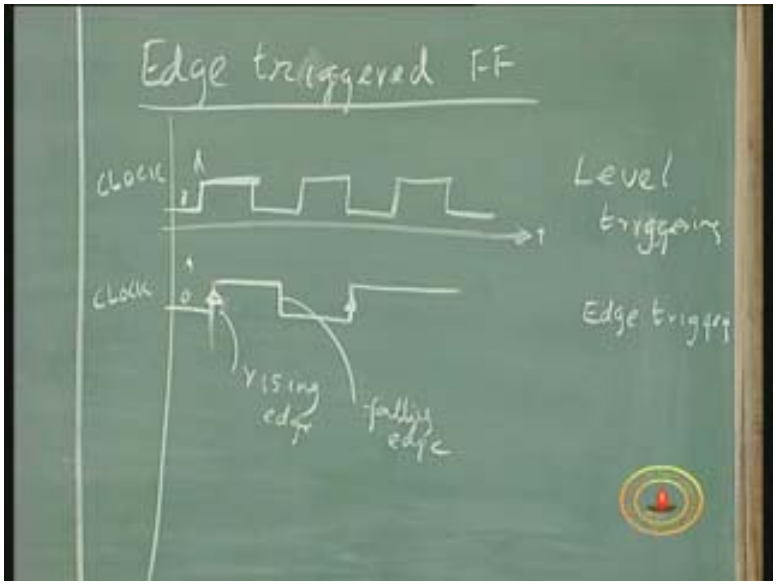
We can make a little bit extra hardware it is not the whole lot of doubling the flip-flop type of thing but some modification in the given flip-flop can lead to a condition that once a change is made it will stay there for the entire duration of the clock and only when the next time the clock changes it will change again. This is where I want to define something called edge triggering and level triggering.

When I said gate, it is an enabling function, gate has an enabling role to play, what did I say, when the gate is high clock is high rather clock is an enabling gate enabling function. Speaking about clock it is an enabling function, we said when clock is high flip-flop works normally and when the clock is low the flip-flop remains in a memory state.

Now within when the clock is high the flip-flop can change the output any number of times. I am not talking about master slave but the normal flip-flop a single flip-flop. If I want to avoid that what I should to do is that, the change in the flip-flop output should not be level sensitive, sensitive to the level of the flip-flop but it should be sensitive to the transition of the gate function. If the gate is high the clock is high, this is clock as the function of time (Refer Slide Time: 39:06) and this is t 0 1 and as long as the gate is high any change in the input will be reflected in the output. I am talking about the single flip-flop but not the master slave remember that. I am making a case for doing a non master slave flip-flop which will do only once in a clock cycle, non master slave flip-flop which will change the output only once in a clock period because I am going to save time delay as well as I am going to save some hardware so I am talking about that.

Therefore when talking about a single flip-flop a non master slave operation when I have this, this is the period during which any change in the input will be reflected in the output. I don’t want this but I want the output to change for only once in that period based on what was the input at a given particular time. This is called level sensitive or level triggering.

(Refer Slide Time 41:53)



On the other hand if I make my flip-flop whatever is the input at the time the input changes from 0 to 1 which is again the clock, the input can be anything here and the input can be anything here but it will recognize the input, it will monitor the input and it will sample the input this is the technical term known as sampling. The flip-flop will sample the input at the time of transition from 0 to 1 and correspondingly decide the output. Any change occurring within that clock period whether it is high or low will not be recognized so output will not change until the next clock transition occurs. Such an operation is called edge triggering operation.

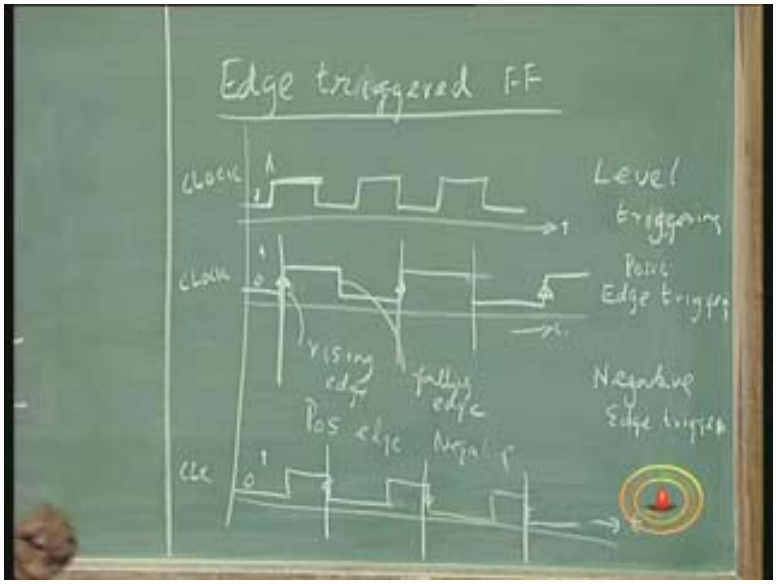
When I say once in a clock period only from 0 to 1 not from 1 to 0 or I can define from 1 to 0 as my reference point and not from 0 to 1 we should not do both the things because I want only one time a clock period I want one change in the output, the output should change once in a clock period, when that's what I want I have to make it either sensitive to the rising edge that is going from 0 to 1 is called rising edge and this is called falling edge. Either the rising edge or the falling edge I can make a reference. There is no need that it has to be a rising edge. Some people call it as positive edge and negative edge. Even though all voltages are positive so we are talking about positive logic as I said 0 level is the 0V and 1 level is some volt like 5V or 3.3V.

When I say positive edge it means going from 0 to 1 and when I say negative edge it goes from 1 to 0. So I can make a flip-flop sensitive to my positive edge alone. Whatever input occurs whatever input happens to be there in the input terminals, whatever signal happens to be in the input terminals at the time of the positive edge transition that will determine the output for that clock period until the next positive edge occurs, that's called positive edge triggered flip-flop. Likewise whatever input happens to be at the input terminals whatever signal happens to be at the input terminals at the time of a negative transition from 1 to 0 only those inputs will determine the output of that flip-flop for the entire

duration of that clock period until the next negative edge occurs. Such a flip-flop is called a negative edge triggered flip-flop.

Therefore for example if the input keeps changing here, again here, again here this called positive edge (Refer Slide Time: 43:38) and on the other hand if the input keeps changing here clock 0 1 t all are functions of time and all these points are called negative edge points and all these points are called positive edge points. So if I design my flip-flop such that input changes will change the output based on of course the condition every time there is a positive transition occurring in clock such a flip-flop is called positive edge triggered flip-flop, input changes will set the outputs every time there is a negative edge occurring in the clock such a flip-flop is called negative edge flip-flop. There we have solved the problem of toggling because toggling can happen only when its level triggered. As long as the clock is high it keeps racing. We have solved the problem of racing because racing occurs when the clock is high and it keeps shuttling back and forth.

(Refer Slide Time: 44:05)



Since you can only change once there is no question of racing. So racing can be avoided by positive edge triggered flip-flop or edge triggered flip-flops plus we don't have two flip-flops like a master slave. That means to that extern I can make it faster the whole flip-flop operation because there is only one delay I have [….45:26] with but of course it doesn't come free that you can't just take a flip-flop and do it but you need to have some extra logic built into your original flip-flop. The original flip-flop as I said is the T flip-flop or a D flip-flop or a JK flip-flop but generally we do it for a D flip-flop.

In a D flip-flop because JK flip-flop is the master slave, most of time JK flip-flop is a master slave flip-flop because there are conditions of feedback and all those extra things. Generally JK flip-flops are master slave flip-flops and D flip-flops are edge triggered flip-flops. The reason is in JK flip-flop the toggling condition is more important they want to use it for toggling conditions and it is doing something meaningfully with

toggling which has an application and such application need not to be very very fast really, it can be fast but need not be very very fast. This is a data storage application D flip-flop a data flip-flop as I said, you put a 1 and it stores, you put 0 and it stores and like to do it as fast as possible as a computer or something like that, an environment where memory has to be accessed very fast, the data has to be put in at a very fast rate and retrieved at a very fast rate. So they usually use edge triggered flip-flop or D flip-flops and master slave operation with JK flip-flop. This is not restricted but generally. You can have master slave D flip-flop no problem.

Now the input can change any time but it will be recognized at the given edge only and the output will change correspondingly. And that extra logic which is required to modify your D flip-flop, the original D flip-flop with the extra modification like some extra gates will block, what really happens is not a very great thing, it blocks the variation because of the nature in which the gates are connected. Once the variation occurs from 0 to 1 it remains at that time unless another transition occurs from 0 to1 so when we look at the circuit we can analyze the circuit and see how this change can occur only once during that clock period. That means the circuit is designed such that only a transition from 0 to 1 in the case of positive edge or 1 to 0 in the case of negative edge can change the inputs of the flip-flop.

Finally the input with SR latch, do you remember that back to back connected NAND gate? It is always there in D flip-flop that is the core, we started with that core, back to back NAND latch, two NAND gates connected back to back. If the input doesn't change for that then nothing can happen. Whether the clock is high or whether the clock is low any change that occurs in the input cannot affect the output as long as the input to those SR flip-flop does not change. In the case of using some extra logic when we need to have some extra gates it will prevent these changes that are occurring except in the case of 0 to 1 transition in the case of positive edge and 1 to 0 transition in the case of negative edge.

I am leaving this as a reading exercise. Please go over some books standard text books in which they will tell you the configuration of edge triggered D latch edge triggered D flip-flop. There is an extra logic involved, please try to understand that logic, please try to first locate that circuit and then understand the operation of that circuit and why the input changes from 0 to 1 and 1 to 0 which can only make a change, the clock changes from 0 to 1 and 1 to 0 can only change the output. That is because as I said it blocks the variation within that period, it blocks the variation to the input of that final latch and the final latch is the SR back to back NAND latch. That latch input should remain constant for that whole clock period whatever it happens elsewhere. Why it happens is that the way in which the logic is designed makes it possible.

I leave it as a reading assignment to you, look at a standard text book in which they will tell you the concept of edge triggered flip-flops, give you a circuit diagram, look at the circuit diagram and understand the circuit diagram of how it works and also analyze why this is behaving the way it is. We have now seen four types of flip-flops as I said D flip-flop, SR flip-flop, T flip-flop, JK flip-flop and using them what we can do we will see in the next lecture.