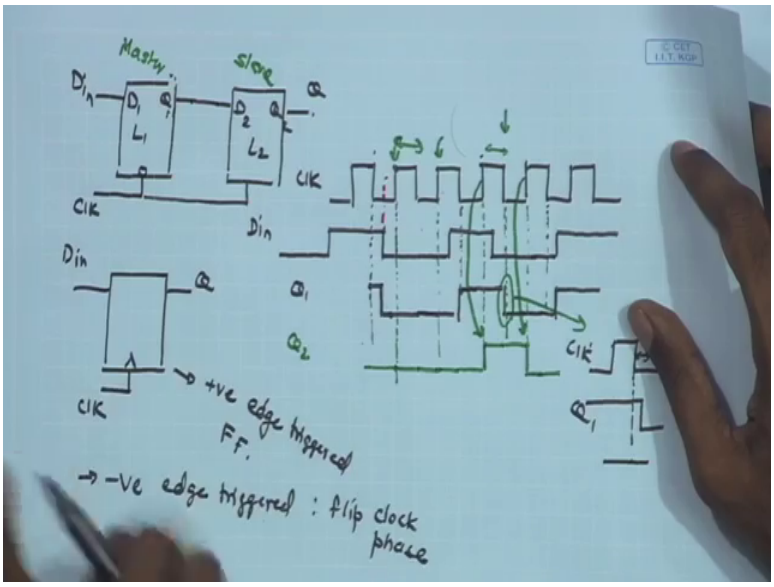
Analog Circuits and Systems through SPICE Simulation
Prof. Mrigank Sharad
Department of Electronics and Electrical Communication Engineering
Indian Institute of Technology, Kharagpur

Lecture – 47
Counter Design

Welcome back to, let us resume our discussion on the flip flops and counter design using these flip flops.

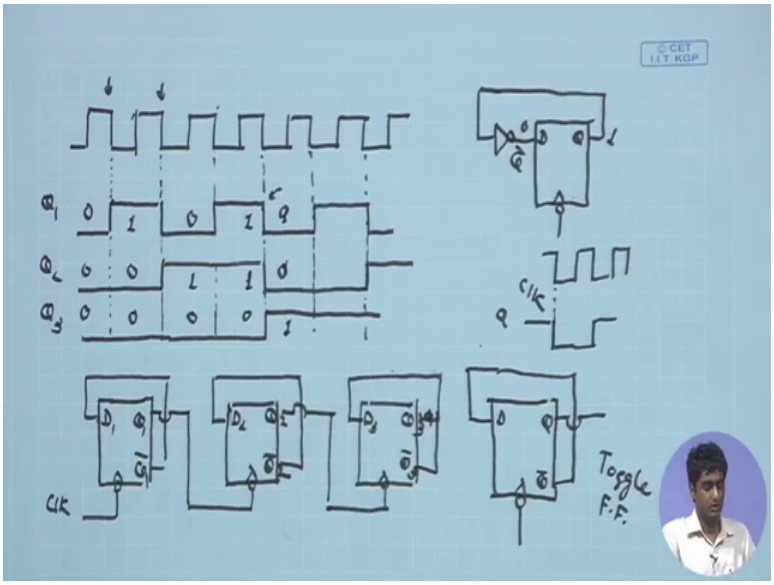
(Refer Slide Time: 00:23)



So, we just discussed the overall operation of level sensitive latch and from there how do we get an edge sensitive flip flop positive and negative edge triggered. From there we can go ahead and try to construct our counter that we discussed in the very beginning. So, once again if I look at the counter operation, I can construct it in 2 fashions one is asynchronous counter which is called ripple counter, and second one is going to be synchronous counter let us see the distinction between these 2 let us first look at a ripple counter which is in asynchronous counter. And you see why do we call it a synchronous or a ripple counter.

So, we will revert back to our truth table for the counting operation and try to implement the overall scheme where each of these flip flop stages in a chain are implementing the division of clock frequency.

(Refer Slide Time: 01:27)



So, once again if I look at the clock waveform and try to see what these how these flip flops are going to be employed to construct the counting operation. Suppose to begin with all the flip flops all the flip flops were set to zero; that means, their outputs was 0. And suppose I assume that all of them are negative edge sensitive; that means, they are going to make transitions at the negative edge of the clock and these points. So, the waveform that I expect is that the first one or the LSB should be toggling at every negative edge of the clock.

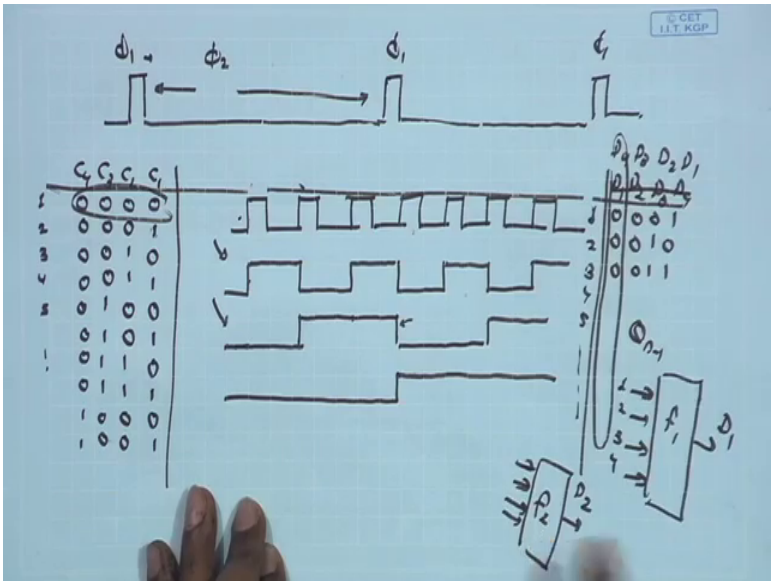
So, you have these as the negative edges of the clock. So, at the this is the expected behavior that I want. So, at the next negative edge of the clock again it should go down at the next negative edge going up and down and so on. And likewise we have the second flip flop that is the second LSB that should be once again transitioning when the first LSB is making a high to low transition. So, if we look at the behavior expected behavior of the second LSB. It should be going high over here and then once again when the next h 8 to low transition in the LSB is happening it should be going down and so on. Once again going up when there is a high to low transition in the first LSB right.

So, if I look at the count that is getting implemented here if it if I call this your having 0 0, after this positive edge of the clock sorry after first negative edge of the clock you are having 1 0, after the second negative edge of the clock you are having 1 0. And then you are having 1 1 and so on. So, if I look at the third one for example, third one once again

is expected to make transition at the negative edge of the second bit. So, if it is 0 starting from 0 it will be going high here and so on. So, after this it becomes 1 1 1 sorry 0 0 1 and so on.

So, this is (Refer Time: 04:28) this is the overall counting operation that you are able to implement from 0 0 0 1 2 3 4 and likewise it will be continuing. So, this is the operation I want from my flip flops. So, it is what I want to implement using the flip flops. So, the first bit that there is a least significant bit of LSB making transitions at the negative edge of the clock starting from all 0 the first flip flop is making transitions at the negative edge going up again at the next positive negative edge going down the next negative edge is going up. So, it is toggling every ultimate negative edge of the clock just like in the truth table we have the LSB toggling after every for every entry in the truth table we have the LSB toggling and the second LSB toggles when we have the LSB toggling from one to zero.

(Refer Slide Time: 05:15)



So, again it goes to 1 1 again the LSB going from 1 to 0, again it goes to work from 1 to 0. So, the second LSB toggles at the down going transition of the first LSB; that means, when the first LSB is going from high to low the second LSB will be making transition. Again here we can see the first LSB going from high to low the second LSB making transition. Likewise the third LSB it is going to make transition when the negative going transition of the second LSB happens for example, if I look at the second one whenever

it is making a transition from high to low that is means a negative going transition of the second LSB it is going to make a transition over here.

So, the third LSB it makes transition when you are having a negative going transition in the second LSB. So, each of these are toggling the state each of them are toggling their states at the negative going transition of the previous stream. So, if I call this say Q 1 Q 2 Q 3, what we can observe is that Q 1 is supposed to make transition at the negative or it is Q 1 supposed to toggle at the negative edge of the clock. Toggle means it is just going to reverse it is state. If it is whatever data input is coming we do not we are not relying on any external data it is just supposed to reverse the state from 0 to 1 1 to 0 0 to 1 1 to 0 whenever the negative edge of the clock is coming. Likewise Q 2 is going to reverse it is state or toggle whenever the negative going transition in Q 1 is happening.

So, if you look at the negative edge of the Q 1 the Q 2 goes from 0 to 1 likewise the second negative edge of Q 1 Q 2 goes from 1 to 0 third negative edge of Q 1 Q 2 goes from 0 to 1. So, whenever negative going edge in Q 1 is appearing Q 2 is making a transition or it is toggling from it is previous state it is changing it is state from 0 to 1 1 to 0 and so on. Likewise same thing for Q 3 Q 3 is going to toggle or change it is state when a negative going edge of Q 2 is appearing. So, here Q 2 is going from 1 to 0 as a result Q 3 is going to make a transition from 0 to 1 it is toggling. So, this is the overall toggling operation that we have to implement using each of these flip flops and make sure that the toggle at that in previous or the negative going transition of the previous output.

So, at this level of course, I can estimate or I can guess that yes the Q 1 can act like the clock for Q 2. So, I can use Q 1 as a clock and increment Q 2 using a negative edge sensitive flip flop. So, that whenever there is a negative going edge or negative transition in Q 1 that acts like a clock for Q 2 and therefore, Q 2 is going to make a transition. At that transition is just reverse. So, if it was 0 before it should go to one it was one before you should go to 0, that is what I expect. Likewise for Q 3 Q 2 can be used as a clock if I use Q 2. If I use a negative edge triggered flip flop to implement Q 3 I can I can try to use Q 2 as the clock for Q 3 now Q 3 being a negative edge sensitive flip flop whenever there is a negative transition in the clock of Q 3, if you are using Q 2 at the clock of Q 3 and there is a negative edge in the Q 2 then the Q 3 should be making a transition. And that transition should be just reversing the state of Q 3. So, Q 3 should be toggling from

0 to 1 that is what we want this is clear definitely we can use Q 1 as a clock for Q 2 Q 2 as a clock for Q 3 and make sure that Q 1 Q 2 Q 3 are all negative edge sensitive.

And the other requirement is that at the negative edge of the clocks these flip flops should be flipping their state, if the state was one it should be going to 0 if the state was 0 should be going to 1. So, we do not need a data external data to increment that. What we need is the information about the previous state of the flip flop itself. If we know that state we just need to make sure that the flip flop is acquiring reverse of that state. So, if the Q 3 output was available to me and it was 0 I want to make sure that the next time at the negative edge of the Q 2 Q 3 is transitioning to 1. So, what how can I implement that? One logical way will be to put the D of Q 3 equal to naught of Q 3 is the previous state right.

So, if I have the latch available where you have the D and Q and you have the negative edge sensitive clock. So, for negative edge sensitive clock will put this triangle to the told that it is edge since it is another bubble to show that it is negative edge sensitive. And I have the Q over here. And if I want to make sure that it toggles the state. So, I would like to I can simply put if I have say an inverter and if I put the Q over here. So, that would make sure that when the Q was say 1 here you had a 0 available, before the negative edge of the clock. So, this is a clock and you are looking at the any of the negative edge of the clock, if Q was one before the negative edge of the clock, we know that the Q bar over here bar means the inversion of Q logical inversion of Q.

So, this was one Q bar was low as a result in the next positive edge of the clock if the Q was high the Q will be making a transition from high to low. Likewise at the next negative edge of the clock the Q 1 or the Q bar will be high because Q is low. So, at the next negative edge of the clock since the D input was high because it being Q bar it will once again make a transition from low to high. So, this is all I need to do to make sure that this flip flop toggles it is state after every negative edge. Is clear? And in general I do not need this inverter explicitly the flip flops do have the complimentary outputs coming in. So, in most circuit interpretation of the flip flop you will have the Q and Q bar the logical inversion of Q bar also available from the flip flop circuit.

So, I can use the Q bar to implement the toggling operation. I can simply connect the Q bar to the D input. And this is going to make sure that at every negative edge of the clock

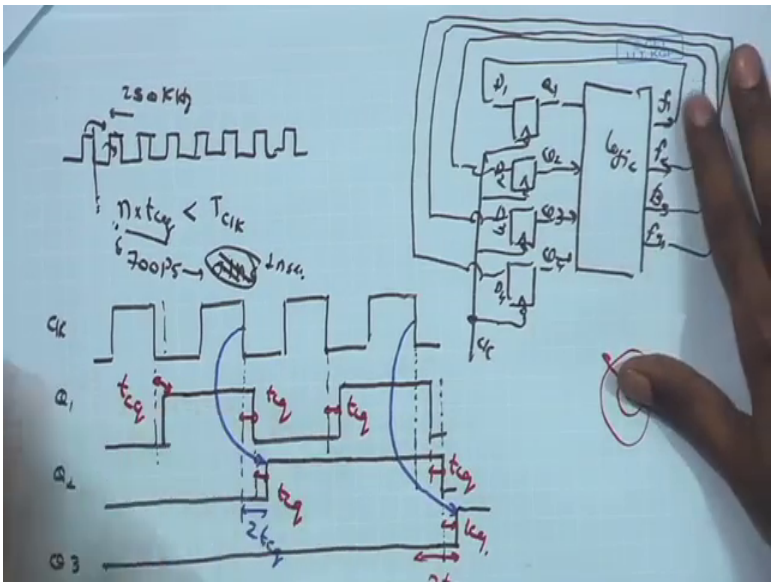
the flip flop output Q is going to flip it is state. From 0 to 1 1 to 0 and so on. So, this can be called as a toggle flip flop toggle flip flop because at every negative edge of the clock is going to toggle it is state from 1 to 0 0 to 1 and so on. Likewise you can have positive edge sensitive toggle flip flop also, here you are using the D flip flop to construct a toggle flip flop which is going to be useful in implementing our counter. So, we have the basic unit for implementing our counter available over here.

And all we need to do in order to implement this functionality. So, cascade such blocks. How do we cascade? The first flip flop in the chain should be receiving the main clock input clock as we said the output of the first clock should be acting like the sorry, output of the first flip flop should be acting like the clock to the next flip flop, So that at the negative edge of the transition of the first flip flop the second flip flop is getting triggered. So, I have these the first one is going to receive the main clock which is going to make this one toggle at every negative edge. And then I would like to put the Q of the previous flip flop as the clock of the next flip flop, So that whenever the if I call this say Q 1 D 1 whenever Q 1 is making a negative transition the Q 2 toggles it state.

So, all I need to do is Q 2 and 2 bar this is your D 2. And once again and we go on repeating the structure and so on, sorry and so on. So, this is what we need to do we need to cascade this structure to obtain an n bit counter. And now we need to answer the question why do we call it a ripple counter. So, if you look at the first flip flop that is receiving the main clock from the system and at the transition of the first flip flop at negative transition the first flip flop the D 2 is changing it is state. Then at the transition of the second flip flop the D 3 is changing it is state and so on. In actual circuit is that we are going to see for implementing this flip flops, there is going to be a delay between the negative transition of the clock and the transition of Q 1.

So, we know that digital circuitry when we are talking about CMOS digital circuitry inverters or any digital logic block they have their own inherent delays. And therefore, if I look at the if I look at the waveforms corresponding to this arrangement, we have the original clock coming over here.

(Refer Slide Time: 16:49)



The Q 1 is going to have a delay with respect to the negative edge of the clock. So, it is going to make transition only after a certain finite amount of delay after the negative edge. So, it is going to toggle after a certain delay we can call this delay at t clock to Q, c to Q there is with respect to the down going edge of the clock what is the time taken by the flip flop to change it is state. So, ideally of course, in our simulation deformation change instantaneously whenever the clock is going low ideal flip flop should transition immediately at the negative edge as per our definition, but because of the inherent delays in the circuitry implementing this flip flop there will be a finite c to Q delay. And as a result it will be transitioning only after a finite delay. Likewise after every edge there will be this finite delay in the transition from high to low transition as well as low to high transition there will be a finite delay.

Likewise if I look at the behavior of Q 3. So, this is your Q 1 this is your main clock and then you have the Q 2. Q 2 how is it going to behave? So, here since it is once again negative level sensitive. So, it is supposed to make transition immediately when the Q 1 goes low, but once again rather than going down exactly at this instance it will have some finite delay if I assume the same delay then again it is going to transition after t cq. Likewise at this point once again it is going to transition after a delayed tcq. So, with respect to this another tcq. So, if I look at the delay of the transition of Q 2 with respect to the original negative edge of the clock, we can see there is going to be 2 tcq delay. So, with respect to the negative edge of the clock you have 2 tcq delay for Q 2. Likewise if I

look for Q 3s waveform it is supposed to make transition at the negative going edge of Q 2.

So, once again we will start from here and stretch all the way and it is going to make transition at the negative edge of the tc the Q 2, but after another tcq delay. And therefore, what is the overall delay that Q 3 is having the transition of Q 3 is having? But the original negative edge of the clock we are having 3 cq delay. So, you have an overall 3 tcq delay likewise if you go on further and deeper into the chain you will have ntcq delay for the n stages of the counter. And therefore, what we can say is with respect to the negative transition of the main clock the outputs are transitioning or rippling from the first output going to the second third 4th and so on.

So, there is a ripple going on just like you drop a stone you know upon we are still what when you drop a stone and then you have a ripples going on step by step. The same thing you are having a negative transition over here and then gradually step by step all are transitioning at the subsequent negative edges after certain delays. So, there is a delay when you put a stone over here and the ripple generates it is taking some time finally, step by step and it is finally, reaching the shore. So, likewise here after the negative transition of the main clock step by step you are having these transitions after each tcq delay finally, after ntcq delay you are having the final output transitioning.

So, this is the concept of ripple counter, all the flip flops are not changing simultaneously at the negative edge of the clock just like we draw here ideally in the ideal waveform this is the picture and we expect that at the negative edge of the clock all the transitions are happening, but in the actual circuitry because of the delays of the incrementing circuitry we will have a rippling effect. So, this is called the ripple counter or asynchronous counter. The other the other flip flops apart from the first flip flop the other flip flops are not strictly edge triggered by the main clock they are triggered by the previous clock. So, asynchronous circuit is one in which a single clock governing the transition for all flip flops.

So, here the main clock is governing the transition of only first flip flop, the other flip flops are not strictly changing their state exactly at the negative transition of the main clock they are changing their state at the transition of the prospective inputs. So, so with respect to the main clock they are asynchronous they are changing as synchronously this

is asynchronous counter or a ripple counter. We can have asynchronous counter as well where we have all the flip flops changing their state simultaneously will briefly discuss that. And that can be useful when your clock frequency required is high and your data rate is high then we may like need to go for asynchronous counter. Before we discussed briefly the synchronous counter which we are not going to use in our design, but just to distinguish it from the synchronous counter we can have a brief very brief discussion. Here how does these delay relate to our original design specification and what should be the limitation on the tcq for our particular application.

So, if you remember the main clock that is coming in our design, we have decided a certain clock frequency for the main clock if you remembered it was (Refer Time: 23:39) 250 kilohertz considering the data rate considering the sampling frequency and the resolution of the bit that we had. The mean we are having around we are having around 2 kilohertz sampling rate and therefore, within 0.5 millisecond we want around one27 pulses of the clock. So, that we can count 127 levels and therefore, we arrive at this 250 kilo hertz requirement for the clock. And we would therefore, like that the ripple should be through within one clock period. So, if it is negative edge sensitive and the data transition stat or the counter starts transitioning from here, before the next negative edge of the clock the oh counter output should be available. Or in other words n times t cq should be less than the period of the clock that we are looking at T clk in this case.

So, n is the number of bits if you are having 7 bit in the counter you are having 7 7 such flip flops, and the output will be coming after ntcq delay. And that should be lower than the clock period that we are having. Now here if we look at the number that we can get from 180 nanometer CMOS technology which you are using in your simulations. We will see that the tcq at the asynchronous circuitry. It can be comfortably within 100 picoseconds a few tens of picosecond 200 picoseconds if you design it aggressively it can be brought down to 10 picosecond. Around one kilo nanometer 100 picosecond can be a safe limit. And therefore, for 7 stages you just have around this number is around 700 picosecond less than nanosecond.

So, this point 1 less than point 1 nanosecond sorry, less than one nanosecond. Another result if I look at the clock period that we have it is 250 kilo hertz, not even one megahertz as a result this one nanosecond delay is going to be comfortable enough for our application, but once again as we target higher number of bits there of course, the

clock period will go on having every bit every single bit increase. And then the delay of the flip flops can become critical component in the design. As if known our application we will see that this delay is not going to be critical, but we should just be aware that if you are trying to extend this concept to different design spec, having higher number of resolution where the required clock frequency is increasing. There of course, at finally, when you go on adding from 7 to 8, 8 to 9 10 11 and so many bits.

Each addition of bit doubles the clock frequency another result you may for higher number of bits you may approach close to the delay of the flip flop. And in those cases the ripple flip flop or the ripple counter may not be are going option. If you are going for a higher number of bits, we may like to go for synchronous counter where we can ensure that all the flip flops are making transition at the same point they are synchronous with the clock whenever the clock is having a transition from high to low all the flip flops are changing exactly at the same time. So, that is the concept of asynchronous counter we are not going to use in this design, but it will just be aware of that concept.

So, very, very briefly we can just discuss the what is the way in which the asynchronous counter is implemented. So, that it leads to the concept of state machine that might have encountered in digital electronics. The state machine basically stores the states of the flip flop stores the states of the logic in the flip flop. So, suppose you have 4 bit counter and you have the outputs Q 1 Q 2 Q 3 Q 4 coming in. So, at a particular clock they are all driven by the same clock. So, suppose they are all negative edge triggered all driven by the same clock. So, the clock is going to all of them, and they are there are the toggle flip flop. So, every clock edge they can change their state. Or I can use the D flip flop I can just have the D flip flop. So, by definition whatever is the D input available over here again that the D 1 D 2 D 3 and D 4.

So, whatever is the D 1 D 2 D 3 D 4 inputs available according to that at the negative going edge the Q 1 Q 2 Q 3 Q 4 will be changing the state. Here I have to make sure that this Q 1 Q 2 Q 3 Q 4 they follow the truth table that I want, you just like the truth table that we discussed the transition in Q 1 p 2 p 3 p 4 should follow this truth table. And for that I need to figure out that if the present state of Q 1 Q 2 Q 3 Q 4 is a certain number of a sudden bit configuration, what is going to be the next state? For that I need to construct a output state for you know given truth table.

So, avoiding the repetition I am just using this truth table all you are drawn. So, if I look at the first second third 4th fifth and you know subsequent cycles. At the first cycle the output of c 1 c 2 c 3 c 4 over all 0. So, what do I expect should be the state 4 c 1 c 2 c 3 c 4 in the next cycle should be? 0 0 0 1, likewise at the second cycle the output of the flip flop is 0 0 0 1, I expect that the output of the flip flop in the next cycle should become 0 0 1 0. Likewise if the output of the flip flops in the third in the second cycle was 0 0 1 0 I would like that in the third cycle it should become So on.

Now how do I make sure that if the output in the previous state or the flip flop output in the previous state was this, how do I make sure that in the next stage the output becomes this? I just need to make sure the D inputs are these right. So, if I have D 1 D 2 D 3 and D 4 I would like to make sure that these D inputs D 1 D 2 D 3 D 4 are these particular values, So that when the next positive edge comes previous state of the flip flop previous values of flip flop output Q 1 Q 2 Q 3 Q 4 where this, if I make that D input such that sorry numbering is just opposite. So, this I should just follow this convention D 2 and D 1.

So, c 1 c 2 c 3 c 4 D 1 D 2 D 3 D 4. So, if I make sure that the D input for the next cycle is 1 0 0 0 then of course, at the next negative edge the clock the flip flops will be changing state the c 1 will be going from 0 to 1 others will be remaining 0. Likewise at the second stage if I make sure that the D inputs are like this 0 0 1 0, previous state of the flip flop or output of the flip flop are 0 0 0 1. Next negative edge of the clock comes and therefore, the D input that were available to all these flip flop will be copied to the output state and another result the flip flop will be turning to 0 0 1 0. So, basically I have to make sure that these are the D inputs this is the weight D inputs are going to change as per the previous values of c.

So, basically how do I implement this I can implement the truth table of each of these D D 1 D 2 D 3 D 4 as a function of the c 1 c 2 c 3 c 4; that means, the previous state depending upon the previous state whatever was the value of c 1 c 2 c 3 c 4 for that I have a dependency of this D 4. So, D 4 is going to have a truth table with respect to the previous state of c 1 c 2 c 3 c 4. Likewise D 3 is going to have a truth table. And each of these can be implemented using a k map you can reduce it for inputs you can conveniently reduce it and try to arrive at the logic which is going to give me the D, the value of D 4 or logic function for D 4 as if a function of the previous outputs.

So, if I have the previous outputs I can call it Q n minus one means previous step output qn minus 1 1 2 3 4 available in the previous state. I will have a logic function f which is going to give me the individual if I call this f 1 this is supposed to give me the logic state D 1. So, that D 1 as a function of previous Q states Q 1 Q 2 Q 3 Q 4 is available, So that I can apply it to the first flip flop input and in the next state according to this D 1 the Q 1 makes a transition. Likewise I will have the logic functions implemented using logic gates for each of these. So, you have the previous state Q 1 Q 2 Q 3 Q 4 this will be incrementing D 2. So, this is just logic function, and then output of these logic functions will be connected to the D 1 D 2 D 3 input over here.

So, basically I have a logic function which is taking these inputs Q 1 Q 2 Q 3 Q 4. And implementing the f 1 f 2 f 3 f 4 truth tables, correspond to logically that is like we discussed and then connecting it back to the D to D 1 D 2 D 3 D 4 and so on. So, this is going to be our synchronous counter. So, only at the negative edge of the clock all the flip plop a changing together. And how are they changing? Thus next state of the flip flop is determined by this D D 1 D 2 D 3 input that is available at the negative edge. And how is this determined that is a determined by the logic function that you have generated using that truth table that we just discussed. So, this is the combinational logic which is incrementing the functions 5 1 5 2 f 1 f 2 f 3 f 4 generating the D inputs for the flip flops and this is a sequential element which is latching that D input and generating the next state. So, this is the sequential element which stores the state or latches the state and gives the input to the logic block the logic block once again produces the f 1 f 2 f 3 f 4 the next state D inputs which should be going to the flip flop and again in the next flip plop negative edge of the clock these D inputs will be lashed by the flip flop generate the Q 1 Q 2 Q 3 Q 4 outputs.

So, this is an example of a state machine where you have a combinational logic implementing functions on the flip flop outputs. So, whatever the flip flop output available at the negative edge of the clock the logic function will take those inputs increment certain functions. And those functions will be fed to or it will be fed to the input of the flip flop. So, that they change state accordingly and you realize a state machine. So, you that you have a desired sequence of state in this flip flop. So, these are these are states of the flip flop right how are we implementing this the state of the flip flop. So, starting from 0 0 0 0 0 1 1 0 1 1 0 these are the 4 flip flop which are

implementing those states how many states in this case if you have 4 bits we have 16 states getting implemented. Once you have 0 0 0 0 what happens again once you have all 1 1 1, but (Refer Time: 35:48) it will be again returning to 0 0 0.

So, it is kind of implementing those 16 states with the help of these flip flops and the logic function. So, this is a way we can look into synchronous flip flop implement a fully synchronous flip flop. And can be suitable for high speed application. Here the delay of course, is going to be constrained by this logic. So, logic also has a delay and you must make sure that of course, if you are going on increasing the number of flip flops the logic is also going to become more complicated. And there we have to make sure that the logic delay is also within limit and making sure that it is able to compute this f 1 f 2 f 3 f 4 before the next edge of the flip flop. But that can be implemented in ways there it goes relatively as a lesser weight as compared to that cq to Q delay c to Q delay, we say that it is going to increase linearly as you go on increasing the number of counters. Or number of bits in the counters the cq Q delay goes on adding linearly, here it is not necessary we can make sure that the logic is faster using some special scheme.

So, there are the 2 schemes we are having fully synchronous counter and then we have the ripple counter, in our design you are going to use the ripple counter. So, let us take a small break and continue our discussion with the counter. And overall control scheme that we are going to generate with the help of this counter.