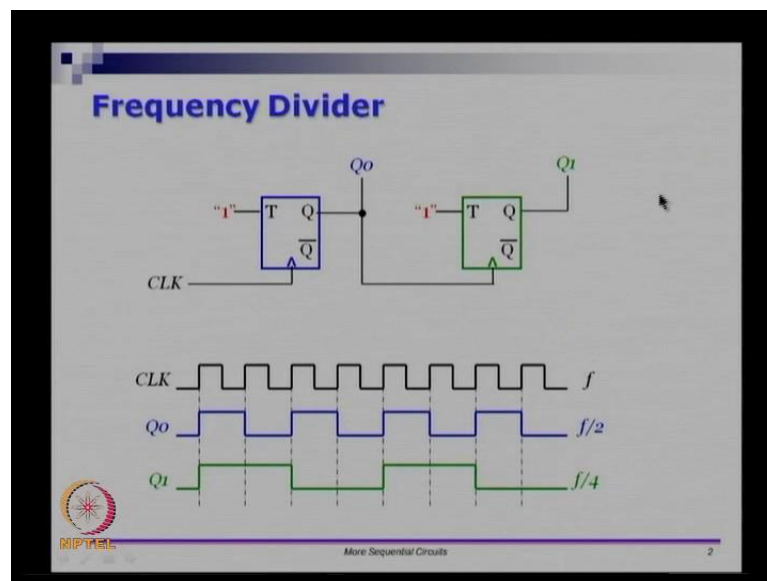


Digital Circuits and Systems
Prof. Shankar Balachandran
Department of Electrical Engineering
Indian Institute of Technology, Bombay
And
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Module - 26
More Sequential Circuits

Hi, welcome to the module twenty 6 of week 5. So, in this module we will take off at where we left in week 4. So, we started talking about sequential elements. And in this module what we will do is we will look at basic sequential circuits. Fairly simple circuits, but many of these are building blocks for other circuits. And we looked at D flip flops, T flip flop, and latches, and so on. So, I want to construct a few circuits using flip flops before we move on to slightly more advanced topics.

(Refer Slide Time: 00:54)



So, let us start with a very simple thing. So, something called a frequency divider. So, let us take a look at the circuit here. We have a T flip flop. And if you think about what would happen if I permanently connect the T input at one for this flip flop. So, I have a clock signal coming in and I am permanently tying the T input to one. If this were to happen, then I would want you to think about what would happen to Q.

So, let us say I give you a clock signal. So, the clock signal is a periodic signal. And it is a T flip flop which is triggered on the positive edge; which means every time the clock

goes from 0 to 1, the output Q_{naught} is supposed to toggle. So, what would happen is if I have, let us say Q_{naught} at 0 initially, this clock pulse would toggle it to one. The next clock pulse will toggle it to 0, the third clock pulse will toggle it back to one, and the 4th clock pulse will toggle it back to 0 and so on. And this would be the wave form that you would get.

The interesting thing about this wave form is that if we look at this wave form and this wave form, you can see that the logical value one is kept for one full cycle and logical 0 value is the output for one full cycle, and then again one and again 0 and so on. So, if you have this signal of clock frequency f or equivalently time period one by f , what you are getting here is you are getting a clock frequency of f by two or equivalently time period two by f .

And therefore we can say that Q_{naught} is a signal that is actually dividing the input clock frequency; divide it by 2. So, Q_{naught} is a signal, which is the clock signal. You take the frequency of it and divide it by 2; that is this clock signal. So, this is a fairly interesting, but a very simple circuit. And something else that you can do with it is you can take this output and connect it as input to another T flip flop.

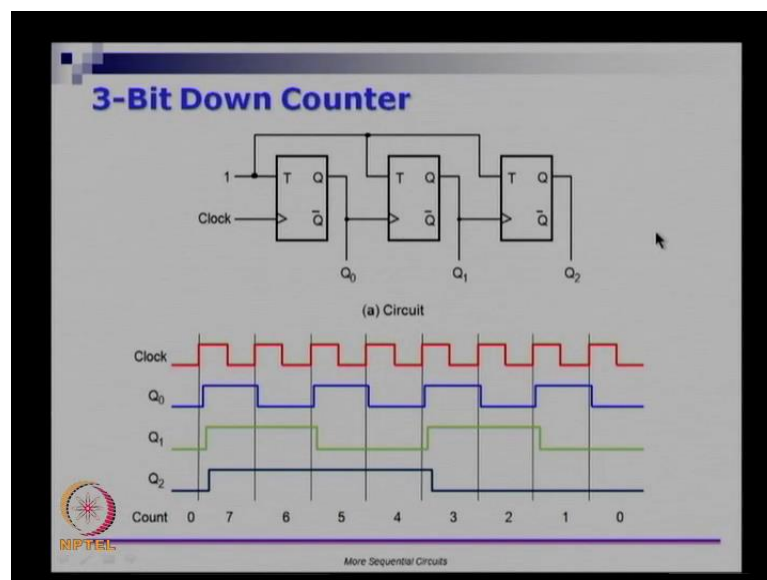
What you think will happen now? So, you have Q_{naught} which is flipping at f by two frequency. So, it is it is switching at frequency f by two. And if you connect to this one, then you can treat this as the clock signal; Q_{naught} as a clock signal and Q_1 as the data signal. And in that case what would have happen is if Q_1 was at one, then when Q_{naught} goes to 0 to one, it will toggle. And when it again goes to 0 to one, it will toggle back, it goes from 0 to one. It toggle up and so on.

Now, you can see that the relationship between Q_1 and Q_{naught} is that Q_1 runs at half the frequency of Q_{naught} . In turn, Q_1 runs at one-fourth the frequency of clock. And this is the fairly interesting thing. Very simple digital circuit can give you something which is very powerful. You get a frequency divider very nicely. So, this is something that can be used in your digital circuits.

So, if you want to run different parts of the circuits at different frequencies, it is possible to actually derive these clocks; which are at, you know the rate of these clocks. So, you can take Q_{naught} and give it to another design, which has to run at half the frequency. Or, you can take Q_1 and give it to design which has to run at one-fourth the frequency and so on.

And with this kind of a setup you can actually design circuits, which can run at f by two power n frequency. So, you can do any division of. So, the frequency f itself does not matter. What you will get from Q_0 and Q_1 is half the frequency, one-fourth the frequency and so on. You attach more and more flip flops like this, you will get one-eighth, one-sixteenth and what not. So, it is a fairly simple circuit, but a very interesting circuits. Then, there are lots of places where you want to count. For example, I want to count the number of electrons that pass through. If I want to do something like that I need a counter. And there are two kinds of counters; up counter and down counter. Up counter counts 0 00, 001 and so on and down counter would count down.

(Refer Slide Time: 05:21)



So, this is the circuit which actually design, which does what is called 3-bit up counter. It is called a 3-bit counter because you have 3 flip flops; Q_0 , Q_1 and Q_2 . And there is a certain connection. There is some method to this ((Refer Time: 05:43)) here or the connection here. And let us see what happens to each of these signals; Q_0 , Q_1 and Q_2 .

So, let us start with the first wave form Q_0 . So, Q_0 if you look at it, it is, there is one permanently connected to its toggle input. And the Q is tapped out. So, you are seeing Q being tapped out. So you will see that Q_0 , like I mentioned in the previous slide, is switching at half the frequency of clock. So, you will see that when clock goes from 0 to one, Q_0 went from 0 to one. And again it went from 0 to one here. So, Q_0 toggles from one to 0 and what not. And that will continue for the rest of the time because you are permanently tied the input at one.

So, one thing to notice here is that we talked about the notion of delay. So, there is going to be a slight delay between the clock's arrival and the Q being ready. So, I talked about; in the previous module, I talked about clock to Q delay for flip flops. So, as soon as the clock comes, Q is not going to be instantaneously available. So, it is going to be available after some period of time. And this delay, I am going to call. This is the clock to Q delay of this T flip flop. So, this is the wave form for Q naught.

Now, let us go and look at what Q 1 has. So, the flip flop Q 1 is also a toggle flip flop. It takes one as its input; which means, it is supposed to toggle for every clock cycle. But the clock that is being fed to this flip flop Q 1 is not the input clock, but it is coming from Q bar. Or, here in this case Q naught bar. Q naught is this signal. So, this Q bar connected here would be Q naught bar.

So, you go and look at Q naught bar. So, this is Q naught and again this flip flop; Q 1 flip flop is actually a positive edge triggered flip flop. But you are feeding Q bar to it or Q naught bar to it. So, you go and look at Q naught bar. And whenever Q naught goes from one to 0, that is, equivalent to Q naught bar going from 0 to one; which is what you are feeding to Q 1. So, whenever this goes from one to 0, this flip flop is triggered and it will toggle its value.

So assuming that Q 1 starts at 0, then this edge, this negative edge of Q naught will be a positive edge for Q naught bar; which will trigger Q 1. So, Q 1 will go and toggle to one. And then it toggles back to 0, it toggles to one and so on. Now, if you go and look at what time Q 1 is toggling, it has to; so clock toggles Q naught, Q naught in turn toggles Q 1.

So, there is a definite time period between the flip flop Q 1's clock input and its output. It is the clock to Q delay of this flip flop. So, if you go and look at the relationship between the clock signal, this red wave form here and the green wave form here, then you have one clock to Q delay that goes and changes Q naught or equivalently Q naught bar.

Let us assume that both Q naught and Q naught bar are available at the same time. Then, you take that and you are connecting it to Q 1. So, that will be another clock to Q delay times away. So, this is one clock to Q unit and this is another one clock to q. So, you have two clocks to Q delays to toggle Q 1. And Q 1 is toggling as shown in the green wave form.

Now if you go and look at Q 2, again it is toggling for its every clock cycle. But the clock is derived from Q 1 bar. You go and look at the negative edge of clock Q 1. And see at these points Q 2 should toggle. So, Q 2 is toggling. And this will be again another clock to Q delay from Q 1's toggle itself. So, overall there are 3 clock to Q delays or 3 times the time for clock to Q delay. If clock to Q delay is let say point one nano seconds, then from the moment this is triggered, it will take point 3 nano seconds for Q 2 to flip.

Now, if you go and read the entries Q 2, Q 1 and Q naught, you can see that Q 2, Q 1 and Q naught is 0 0 0 or count 0; 0 0 1 or count one; 0 1 0 or count two and so on. It goes up to 1 1 1. So, Q 2 is 1; Q 1 is 1; Q naught is 1. So, from bottom to top it is 1 1 1 one; that is 7. But something interesting is happening after that. You have each of them at one. And each of them is going to toggle the next time around because this clock, this red wave form, that clock is coming in; the blue wave form will toggle, because the blue wave form is going from one to 0. So, Q naught is going from 1 to 0, which means Q 1 will toggle and Q 1 is going from one to 0. So, Q 1 bar is going from 0 to one. So, Q 2 will also toggle. So, what you are seeing is you are seeing 0 0 0 here. So, we counted 0, 1, 2, 3, 4, 5, 6, 7.

In every clock cycle, we have one unit and it goes to 0. And the interesting thing here is the moment you get 0, the behavior is going to be similar to as though we are starting from 0 here. The only thing is we have this clock to Q delay that got added up and that was not here in this part. So, this will keep cycling as 0, one, two, 3, 4, 5, 6, 7; 0, one, two, 3, 4, 5, 6, 7 and so on. And this is called a up counter.

So, the reason why it is not counting to infinity is that we have only 3 flip flop. So, this is just like the speedometer that I have been taking about for a long time. So, if you have a speedometer that has only 3 digits. It will go from 0 0 0. It will go all the way after 9 9 9 and it will go back to 0 0 0. You have no way to say that it is thousand, once it rolls over. So, the same thing is happening here. It is rolling from 0 0 0 to 1 1 1 and it rolls back to 0. So, this is called a up counter.

Equivalently, we have something called a down counter; a 3-bit down counter is very very minor change. You take the same 3 flip flops and connect them. Connect all the toggles to one. This time instead of connecting Q bar to the clock input, connect Q to the clock input. So, we are connecting this Q naught as that clock for Q 1. Q 1 as the clock for Q 2 and so on. And you tap out Q naught, Q 1, Q 2. Again if you go and look at, what

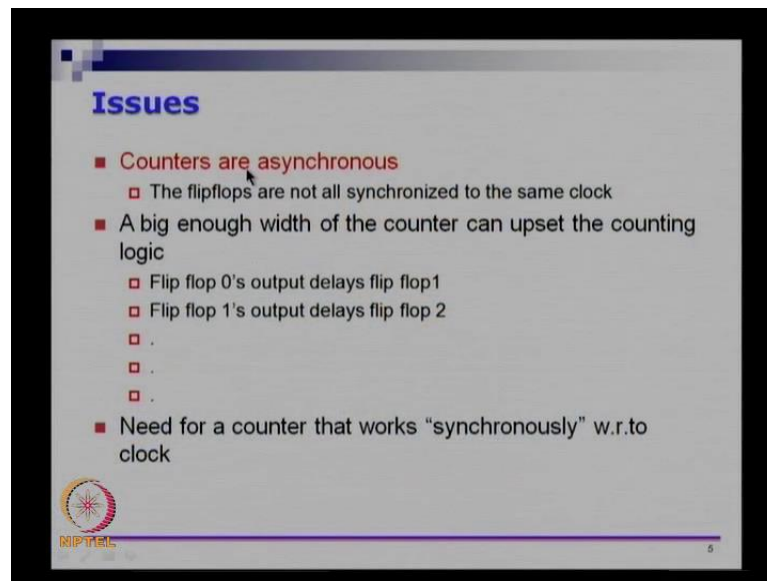
would happen? You would see that clock is running at whatever frequency it is. Q naught will toggle every time the clock pulse comes in. So, you have Q naught which is toggling, every time the clock pulse comes in. So, in all. So, in this clock pulse, this one is toggling to one, and these clocks toggles back to 0 and so on. And since we have connected Q naught as an input to Q 1 as its clock, so this flip flop Q 1 has Q naught as its clock, then Q 1 toggles from 0 to one whenever Q naught goes from 0 to one.

Similarly, when Q 1 goes from 0 to one, Q 2 also goes from 0 to 1. Except that there is a small delay. So, going from clock to Q naught is a one clock to Q delay, again another clock to Q delay and another clock to Q delay. So, this will take 3 clock Q delay units. And this is the basic setup.

Now, if you go and read what is going to happen? Again Q 2, Q 1, Q naught, right. 7; so this 1 1 1 is 7; then you have 1 1 0, which is 6; then you have 1 0 1, which is 5. So, it goes all the way down to 0. All the way down to one. So, you have 0 0 1 and then it is goes back to 0. And here, we started with 0.

So, you can imagine that if this is running forever, then this 0 will then toggle to 7 and so on. So, if you go and read the values of Q 2, Q 1, Q naught, it is going in the descending order. And this descending order is essentially the down counter. You are counting down. And as before when you count from; let us say if it is a 3-bit value, it counts from 7 down to 0 and it goes back to 7. So, there is; it is actually an increment going from 0 to 7, but then because we do not have enough bits; that is what can happen. So, you will count from 7, 6, 5, 4, 3 up to 0 and back to 7. If you give one more bit here, then you will start with 1 1 1 1; which is fifteen. You will go from fifteen down to 0 and then you will go back to fifteen. So, you can add as many flip flops as you want to the right side. And you should read the number from right side to left side; that will give you the down counter. So, we have discussed two counters; up counter and down counter.

(Refer Slide Time: 15:19)



The slide is titled "Issues" in blue text. It contains a bulleted list of issues with asynchronous counters. The first bullet is "Counters are asynchronous" in red, with a sub-bullet "The flipflops are not all synchronized to the same clock". The second bullet is "A big enough width of the counter can upset the counting logic", with sub-bullets: "Flip flop 0's output delays flip flop 1", "Flip flop 1's output delays flip flop 2", and three empty square boxes. The third bullet is "Need for a counter that works 'synchronously' w.r.to clock". At the bottom left is the NPTEL logo, and at the bottom right is a small number 5.

- Counters are asynchronous
 - The flipflops are not all synchronized to the same clock
- A big enough width of the counter can upset the counting logic
 - Flip flop 0's output delays flip flop 1
 - Flip flop 1's output delays flip flop 2
 - .
 - .
 - .
- Need for a counter that works "synchronously" w.r.to clock

But, there is a small issue with both these counters. The counters are what are called asynchronous. The flip flops are not all synchronized to the same clock. So, imagine I have a big enough width of a counter. So, I have a 100 bit counter; which means, it can count from two power, sorry, it can count from 0 to 2 power hundred minus one. Let us say something like that.

A big enough counter can have, can encounter a problem because flip flop 0's output is going to delay flip flop one. Flip flop one can toggle only when flip flop 0 goes. The output of flip flop 0 is ready. So, there is a clock to Q delay from the input clock to flip flop 0. There is a delay because of flip flop one. Its effect is seen in flip flop two. The effect of flip flop two is seen on flip flop 3 and so on.

So, if I have hundred flip flop which are in a chain, you will see that hundred times the clock to Q delay. Let us say it is point one nano seconds. Hundred times point one would be ten nano seconds. So, you will see that the hundredth flip flop is actually flipping ten nano seconds after the clock pulse comes in. And if your clock width is greater than ten nano seconds, you still getting all of that done in one clock cycle. But what could happen is if your clock width is, let say less than hundred nano seconds, let say it is 75 nano seconds, then what would happen is your counter is actually counting at a rate which is beyond the clock width. So far going from one count to the other in the next clock pulse, you would want all the count; all the bits in the counter to settle down to a value. You would want; if it starts from 0, you would want one. But this will not happen. What

would happen is in one clock cycle, some of the flip flops will settle down to the count values. But some would not have. And this is a problem.

So, for an external observer who is going to take the values for every 75 nano seconds, they may not read the correct values out. So, what we need is we want all the flip flops to work based on the clock. So, the clock should be the triggering signal. And based on the clock that comes in, all the flip flop should settle down to their respect values within one clock cycle. And such a circuit would be called as synchronous circuit. What we want is each of this flip flops to change in such a way that their output values go down to their final values within one clock cycle.

(Refer Slide Time: 18:00)

Observation

Clock cycle	Q ₂	Q ₁	Q ₀
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1
8	0	0	0

Q₀ toggles every clock cycle
 Q₁ toggles every time Q₀ changes from 1 to 0
 Q₂ toggles every time Q₁ is at 1 and Q₀ changes from 1 to 0

Q₁ changes
 Q₂ changes

More Sequential Circuits

So, let us look at the 3 bits. So, let us go and design a synchronous circuit. Let us see how to do that. So, what we want is we want in clock cycle 0, one, two, 3, 4, 5, 6, 7, and eight and so on. We want to go from 0 0 0 to 0 0 1 to 0 1 0 to 0 1 1 and so on, all the way up to 1 1 1 and again go back to 0 0 0. This is what we want. If we go and look at what is happening at Q naught, Q naught is toggling every clock cycle. That is fine. Now, you go and observe Q 1. So, Q 1 is changing here, here, here and here; Q 2 is changing here and here. These are the changes that you see in Q naught, Q 1 and Q 2. This is not something that is new to you.

Now, let us see how to derive some synchronous circuits. Synchronous circuit should mean that you should take a single clock pulse. And your input should be in such a way that the output can be settled in just one clock cycle. So, let us look at this. So, Q naught

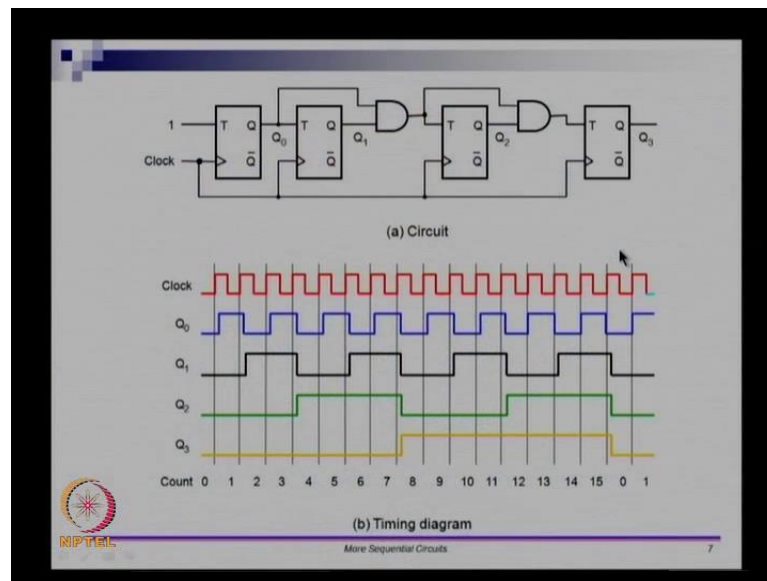
is toggling every clock cycle. So, this is fairly easy. We saw this earlier. And if you have just one bit, then there is nothing. So, it is anyway taking the clock input. So, it is actually synchronous with respect to the clock. If you go and look at Q 1, Q 1 is toggling every time. Q naught changes from 1 to 0. So, you go and look at this Q naught as changing one to 0. So, Q 1 toggles, Q naught is going from one to 0, Q 1 toggles; again one to 0, Q 1 toggles. This is again going from one to 0, Q 1 toggles. So, Q 1 toggles every time, Q naught changes from 1 to 0.

You go and look at when Q 2 is toggling. So, Q 2 is toggling every time Q naught is. So, Q 1 is one and Q naught changes from one to 0. So, you see. So, let us take this little slowly. Q 2 is; at this point, it is a 0. So, and it has toggled to one in clock cycle 4. And, what are the conditions, that is making it toggling? You go and look at clock cycle 3. At the end of clock cycle 3, we have Q 1 is 1. And in the subsequent clock cycle, Q naught is going to go from one to 0. So, whenever Q naught goes from one to 0 and Q 1 is one, Q 2 is toggling. Let us validate this statement.

So, let us go and look at the second place where Q 2 toggled. It is going. It is toggling here. So, I said the statement. I said was whenever Q 1 is one and Q naught toggles from one to 0, Q 2 will also toggle. That is the statement here. So, we are seeing that. Q 1 is at one, Q naught is toggling from one to 0. So, Q 2 seems to toggle from 1 to 0. So, and you will see that this is going to happen one after the other.

So, this is an interesting observation. Now, I want you to take a break and think about when will Q 3 toggle. So, go and think about it and solve this on a piece of paper before you come back to the video. So, my question is when should Q 3 toggle. So, I am going to give you the answer in the next slide. But I want you to take a minute and think about it.

(Refer Slide Time: 21:29)



So, I have given a circuit here, which actually has a 4-bit counter. And you can see that Q₀ is toggling every clock cycle; Q₁ is toggling whenever Q₀ is toggling from 1 to 0. So, you can see the input. It is; whenever Q₁ is at one, this will toggle. And for Q₂, the condition that we derived was Q₁ must be one and Q₀ must be going from 1 to 0. So, you look at this. Q₁ must be one. And if Q₀ is currently one, we know that it is going to switch to 0 in next clock cycle. We do not have to wait for this trigger to go from one to 0. If Q₁ is one and if Q₀ is one, we know that Q₀ is going to go to 0 in the next clock cycle. So, I do not have to look for this switch from one to 0. Instead, I will go and look for the value which is one. So, that is what is done here.

You go and look at Q₁ being one. And if Q₀ is one here, in the next clock pulse you know that it is going to go to 0. So, which means that is a transition from one to 0 on Q₀. And however the current value of Q₁ is one and Q₀ is one, these two conditions is enough to go and toggle Q₂. And the answer to my question in the previous slide is that if Q₂ is one and if Q₀ is one; so and if Q₀ and Q₁, they both go to from one to 0, then you will have Q₂ toggling. So, Q₃ toggling. So, look at Q₃ is input. Q₃ will toggle when Q₂ is one as well as Q₁ is one as well as Q₀ is one. If all 3 are one, then this will toggle. So, it is actually a fairly simple circuit now. So, the condition is for any nth bit to toggle, then you have to have a logical AND of all the previous bits. If they are all one, then this bit will flip.

So, if you want to get a counter with 5 bits, you have Q₀, Q₁, Q₂, Q₃. For Q₄, you take this input. And with Q₃, give that as toggle input to flip flop Q₄. That will be a

5-bit counter. So, these are the wave forms that you are seeing here. So, clock is at whatever clock frequency it is, if you go and look at when Q naught will switch. Let us look at the delay between Q naught and its clock. So, Q naught is going to flip a certain time unit away from the clock. So, that is one clock to Q delay. So, you have one clock to Q delay.

Then, if we go and look at when Q 1 will toggle, so you have Q naught. So, Q naught is going from 0 to one. And Q 1 will toggle if the input was already one. So, if this is one, then if the flip flop Q 1's setup time is satisfied. In this case, it should be because Q 1 is here. Q 1 clock pulse is coming here. It is a same clock that is going. You see this is now a clock. The same clock is going everywhere.

So with respect to this clock, you go and look at its input. The input is Q naught. Has it settled down? Yes, it has settled down; which means Q 1 can flip in clock to Q delay away from this clock, not from Q naught. So, Q naught is data to Q 1 and clock is the input clock signal itself. So, the triggering signal is clock. The sample signal is Q naught. The sample signal Q naught remains at one, when the triggering signal clock is coming in. So, as long as the triggering signal comes in, only when the sample signal is stable, then the set up time would get satisfied. So, it is been stable for almost one clock period. And for a good flip flop this should be enough. So, this toggles. The same thing happens for toggling down also.

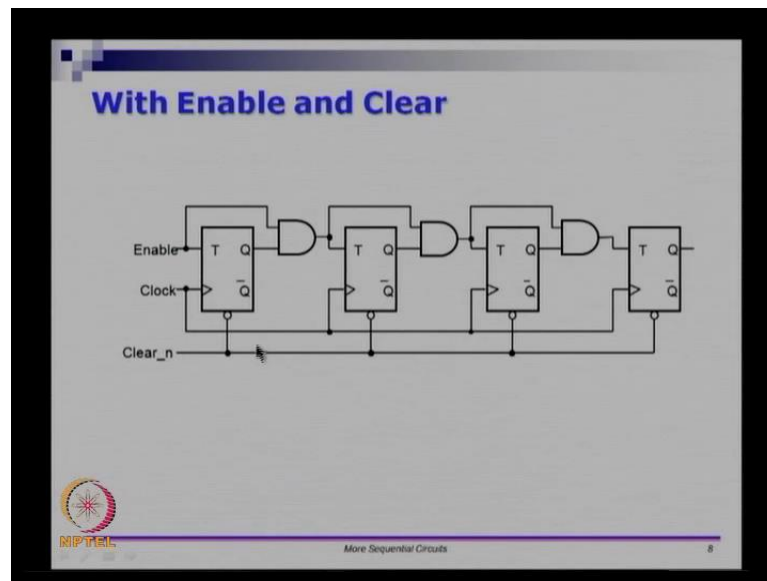
So, you go and look at this place where Q 1 is toggling from one to 0. So, the clock pulse is this. And you go and look at sampling of this wave form. So, Q naught is at a stable one. Q naught is a stable one. And Q 1; the previous value of Q 1 is one. So because of which it can go down to 0. So, this happens from with respect to Q 1, Q 2 is toggling. And with respect to Q 2, Q 3 is toggling. But the clock signal that is given is the same clock pulse. So, what you are seeing essentially is the delay. The delay is measured away from the clock and each flip flop has a certain time unit. The clock to Q delay; that it is required for the output of the flip flop to switch. The inputs, however are ready when the clock pulse comes in. so all the inputs are remaining stable. You go and look at this condition. Q 1 is one; Q 2 is one and Q naught is one, when this clock pulse comes in. Since all 3 are stable, this input will be one. So, it will take AND gate delay, whatever the delay of the AND gate takes and so on.

Eventually, the AND gate will settle down. We will hope that it is going to get settle down before the clock pulse comes in. And when the clock pulse comes in, all of these are stable. The AND gate will have a stable value as a output; which means a toggle input is seeing a stable input. So, then Q 3 can toggle; because it is sampling. So, this one is sampling t. So, when it samples t, the output should toggle in this flip flop. So, it is toggling. So, you can notice that the delays are not adding up. From clock, all the Q bits are only one clock to Q delay away.

So, then what is hidden here is what is not shown in the wave forms here is that there is a delay, but the delay is because of these AND gates. So, if we keep adding more and more flip flops like this, if you go to Q 4, Q 5, Q 6 and so on, what would happen is you would add one AND gate to each one of these. So, eventually what can happen is for a wide enough width of the counter, you have several AND gates that you have to connect for the input to be; for the toggle line. So, at the n th stage of the flip flop you will have n minus one AND gates. If this n minus one AND gates, if the delay of that is more than the clock width, then you will actually violate the setup time of the n th flip flop. So, that what will happen? And the moment you violate the setup time that flip flop can go into meta stability. So, this is a concern, but this circuit is called as synchronous circuit.

So, there are several ways in which you can cut down on the delay. You do not have to wait for n minus one AND gates delay. You can do better things than that. But as of now you can see that all the outputs are actually just clock to Q delay. Each of the bits are only a clock to Q delay away from the clock. They are not adding up. The delays from the clock are not adding up. There are a few variations that you can look at.

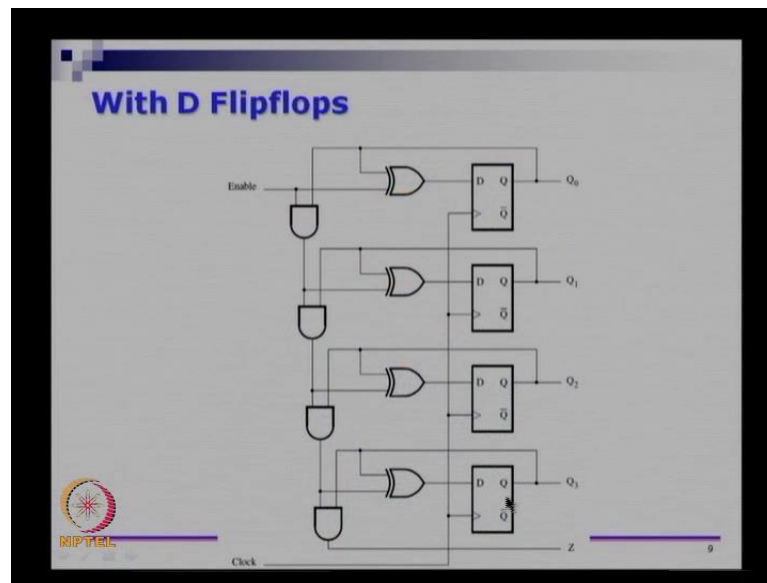
(Refer Slide Time: 29:05)



So, this is the circuit which has both an enable and a clear. So, there is a enable line that is given as an input. And that is ANDed at each of the stages. And there is a clear bar line, which is going into negative clear line. So, we will come to the clear part later.

Let us look at the enable part. If enable is one, then this T is one. And earlier we had this and anyway. So, that will remain the same. However if enable is 0, all the toggle inputs will be at 0; which means, the counter will not count. So, this is not going to count. The clear signal; if clear bar is 0, then clear is one. So, if clear bar is 0, then clear is one. This will go and clear all the flip flops. however if clear bar is one, then this will be 0; which means, the counter will count as though it is a regular counter. So, this is the very fairly simple circuit, which has enable added and clear added. You can also design these counters with D flip flops. In this case, we have a flip flop Q 3, Q 2, Q 1, Q naught. We have a single clock. So, I do not know whether you can see it clearly.

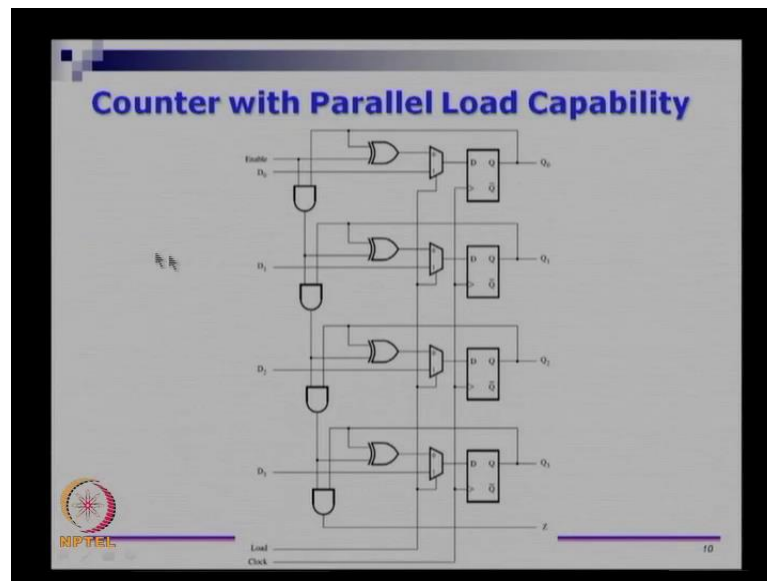
(Refer Slide Time: 30:14)



But there is a clock signal here at the bottom that is going into the clock input of all of these. There is an enable line that is coming in. And there is a nice self-repeating structure. So, Q_0 is XORed with enable that goes as D input here. Then Q_1 is XORed with enable and Q_0 . And that is going as D input here. Then Q_2 is XORed with Q_1 and Q_0 and enable.

So, if you go and look at this part of the circuit, it is enable and Q_1 and Q_0 . If you have all 3 that is XORed with Q_2 itself. And that is going to be the input to this flip flop and so on. So, this enable circuit is as it was in the D flip flop, in the T flip flop structure. So, you go and convince yourself that putting an XOR gate will actually let you do this counting. So, in this case this is an up counter. You go and convince yourself that these are the values; that Q_0 , Q_2 , Q_1 , Q_2 , Q_3 , what are they going to have when assumed that all of them are initialized to 0. Then go and see what would happen to each of these bits as and when every single clock comes in. So, if you want keep it simple, go and look at Q_0 initially, then add Q_1 and look at how it is counting. Then go and look at what Q_2 is doing and see how it is counting. And you should see that it is working like a counter. You can take this circuit and you can also add a parallel load capability.

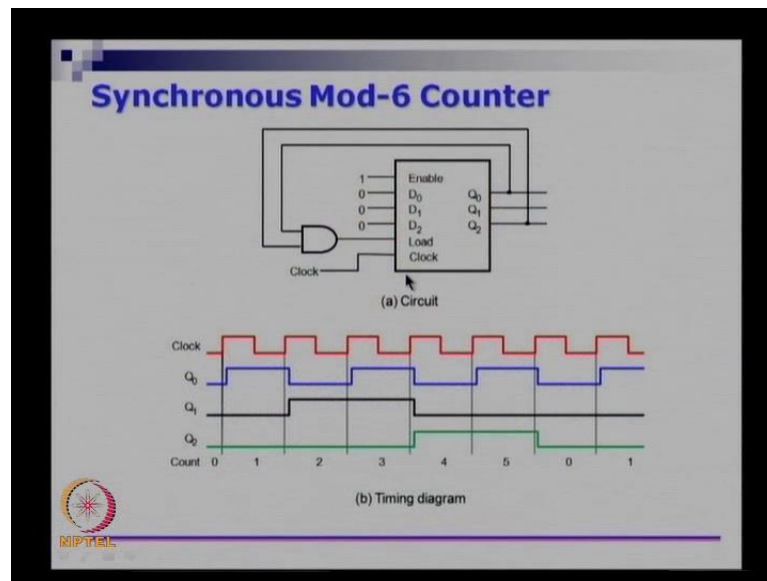
(Refer Slide Time: 32:01)



So, what we have done to the previous circuit is you see the AND gates structure and you see the XOR gate structure. They are remaining as they are. But there is a mux that is added before it goes to the D flip flop. And what is this mux taking? It is taking a select line, which is called a load line. So, whenever load is one, it is taking an external input. You can see that it is true for each one of them. So, there is D naught, D one, D two, D 3. All of them are taking external input. Like these are all external inputs. They are going to the mux. So, if load is one, then the external input is placed as D inputs for all these flip flops. And that will become the starting value of the counter. If load is 0, this will work like a counter. So, what you are essentially having from this circuit is that if load is one, you can load a value to the counter and then you make load equal to 0. And from there it will count.

So, I can count. So, I can load a value 5 by making load equal to one and by putting 0 1 0 1. Here, I can load a value 5 and it will start counting from 5, instead of counting from 0. So, it will start from 5, then go 6, go to 7, go to 0, go back to one, two, 3 and so on. You can start the counting at any point of time and load a new value. So, maybe I am going from 5 to 6 to 7 to 0, then I want to load 4. I can bring load signal by placing the value 0 1 0 0 here and bring the load signal. It will inhibit. It will prohibit counting up. However, it will take the load value and count from 4 onwards. So, again I suggest that you go and look at how this works by drawing wave forms and convincing yourself that this circuit will work. Then, you can do a few other things.

(Refer Slide Time: 33:52)



So, this is a circuit which a synchronous mod-six counting. So, there is a clock signal and this has an enable signal, it has a load signal. And just like what we did in the previous one. So, let us take something like this. This will always count from 0, one, two, 3 and so on up to fifteen and then go back to 0. So, I may not want to go to two power n minus one every time. So, if I use a 3-bit counter, it will go from 0 to 7, then go to 0 and so on. What is if I want a mod 6 counter? a mod 6 counter goes from 0, 1, 2, 3, 4, up to 5 and then goes back to 0, because a number modulo 6. So, 6 modulo 6 is 0. So, it goes to 0. Then from 0, again it goes from 1, 2, 3, 4, 5 and again it goes back to 0 and so on. That is called a synchronous mod 6 counter.

So, that is called a mod-six counter. And this circuit is a synchronous mod-six counter; because we have taken the previous one. This circuit is here without the last stage, without the D 3 stage. If we take the top part of the circuit and we do a small check, what we are doing is we are making enable equal to one always. And we are going to load 0 0 0. The condition for load is that if I take 5, the moment I hit 5, the next clock cycle I want to go to 0. So, 5 is 1 0 1; which means Q₂ should be one, Q₁ should be 0 and Q₀ should be one. You take that condition when Q₂ is one and when Q₀ is one, you and it together. This means, this line will be one only when the counter has reached 5. The moment you have reached 5, then this and will have a one. That is coming in as a load for this counter. So, the moment load comes in from 5, it will not count to 6. It will load whatever is at the input. So, D₀ is 0; D₁ is one; D₂ is 0; that will get loaded as the counter value.

So Q_0 , Q_1 , Q_2 will go to 0. So, that is what you are seeing here. So, it goes from 0, one, two, 3, 4 up to 5. When you hit 5, this AND gate will have 0. So, till then it will be 0. You can verify that. So, till you hit here, Q_1 is this AND gate's output is 0. But it is this point; Q_1 is one and it will load the current set of values. So, there is also asynchronous version of this, which have given in these slides. I suggest that you go and read through the asynchronous version and see how it works. So, this brings me to the end of these sequential circuits.

In the next module, we will see the basic sequential circuits which are slightly more complicated than the counters and other things that we have right now. So, I will see you in the next module and bye, bye.