

Digital System Design
Professor Neeraj Goel
Department of Computer Science Engineering
Indian Institute of Technology Ropar
Lecture 44
Counters

Hello everyone, now based on whatever we have studied in our previous lecture, that we like we have studied in a previous lecture different kinds of flip flops, RS flip flop, JK flip flop, D flip flop. Based on them, we will try to see how we can utilize them to create different kinds of utilities.

(Refer Slide Time: 0:38)

Flip-flops



- Type of flip-flop
 - SR, D, JK, T FF
- Can we design one flip-flop from another?
- Counters
- Sequence generators



Digital Logic Design: Sequential Circuits.

The first thing is that can we convert, so can we use one flip flop to design another. So, let us say we have SR flip flop, we know that from SR flip flop we can design D flip flop, but can we design JK flip flop or D flip flop using D flip flop or basically one flip flop from any other flip flop.

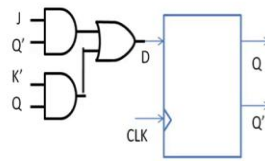
Then in this lecture, we will also see that how these flip flops, let us say D flip flop, how it can be used to design clock divider, counters or sequence generators. The idea of this lecture is that various applications where these flip flops can be used, and rest is left on to imagination that were all we can use these flip flops or memory elements. More particularly synchronous flip flops.

(Refer Slide Time: 1:43)



Design JK FF from D-FF

J	K	Q	Q'
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0



$$Q^+ = JQ' + K'Q$$

$$Q^+ = D \text{ (for DFF)}$$

$$D = JQ' + K'Q$$

Digital Logic Design-Sequential Circuits.



So, let us say we would like to design a JK flip flop from a D flip flop from the theory of JK flip flop, we know that this is the truth table of JK flip flop that when J and K both are 0, Q plus means the next state of the Q will be same as the previous Q if J is 0, K is 1 next state is always 0 independent of the previous state, when J is 1, K is 0 the next state is 1 irrespective of whatever was the previous state, when J and K both are 1, then the next state is inverse of whatever was a previous one.

So, if Q was 0 then Q plus next state of Q is going to be 1. If it was 1, then next state is going to be 0. So, we can write all of these in form of we can solve K map and we can see that Q plus can also be written as J Q dash plus K dash Q. Now, we would like to design this JK flip flop using a D flip flop.

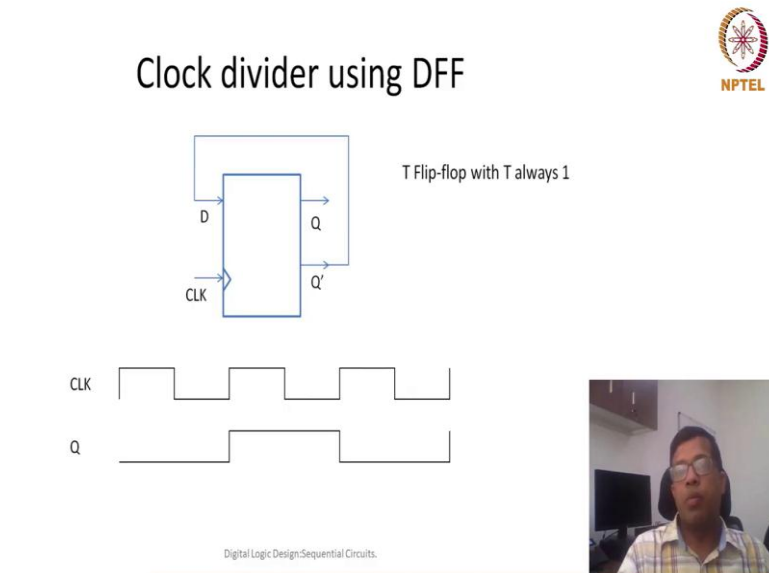
So, D flip flop is a delay flip flop, whatever is the input of D that would be represented by Q at the active edge of the clock. So, we can also write that Q plus is equal to D for D flip flop. Now, if this is so, then we can replace this Q plus this we can whatever is the value of this J Q dash plus K dash Q, we can assign it to D and then essentially this whole system will work like a JK flip flop.

So, if I take J and K as an input and provide this J and K dash as input to this NAND gates and Q and Q dash I connect it from these, this output of my D flip flop, then essentially this whole system represents a JK flip flop. Although underneath I am using D flip flop, but the functionality of this whole circuit. So, if I say that this this become the complete circuit, where J and K is the input another input is the clock and the output is Q and Q dash, then this system is actually a JK flip flop.

Similarly, by looking at the characteristic equations of two flip flops, then we can think of designing any flip flop from the other flip flop. So, this is one of the interesting applications of our like flip flops that once we have one then we can also try the other one.



(Refer Slide Time: 4:44)

Clock divider using DFF



The diagram shows a D flip-flop with inputs D and CLK, and outputs Q and Q'. A feedback loop connects Q' to D. The text "T Flip-flop with T always 1" is next to it. Below the diagram is a timing diagram with two waveforms: CLK (clock) and Q (output). The CLK signal is a square wave. The Q signal is a square wave that changes state at every rising edge of the CLK signal, resulting in a frequency that is half of the CLK signal.

Digital Logic Design: Sequential Circuits.



Now, let us look at the other application. So, can we design a clock divider using D flip flop. So, rather than looking at so this is my D flip flop so rather than looking at design, what I am trying to show you is the direct implementation Then we can analyze that how it is working. So, if this is the, this is my D flip flop, and I know that whenever clock, so whatever is, given the clock was the edge of the clock, this D would be transferred to Q.

So, if I make sure that the at every clock edge, my D is the alternate 0 and 1 0 and 1, then this will make a new clock, which would be half the frequency. So, for that what I need to do is because if D is whatever if the Q is actually equal to D, but Q dash is equal to node of D. So, that means if I can connect this Q dash to D, so then I would have an alternative positive and negative values at the value at output Q.

So, let us understand it with, so basically I am connecting this Q dash with the D. So, let us understand it with this clock signal, so with the input clock waveforms. So, let us say this is my regular clock, now at the regular clock, let us say initial output was 0. So, if initial output was 0, then Q dash is going to be 1 all the time and if it is 1, then at the positive edge of this clock, the next output is going to be 1 and for how long it will remain 1 it will remain 1 for whole 1 clock duration.

Similarly, at this moment, when Q was 1, Q dash was 0, because Q dash was 0 at the next positive edge, the output Q will become 0. So, this way, my Q would keep on alternating and it also looks like a clock signal, but of half the frequency or double the clock period. So, in the in this way my D flip flop can be effectively utilized to home as a frequency divider here we have seen it can be used to divide the frequency by 2.

So, whatever is the clock frequency, the output Q here represent clock frequency f by 2 or the time period is a double of previous time period, duty cycle is in this case it is same 50 percent. So, whatever is the, like whatever is the duty cycle of initial clock, it does not matter, but the duty cycle of Q is always going to be 50 percent because they are like it is half it is low for one clock period and again high for one clock period.

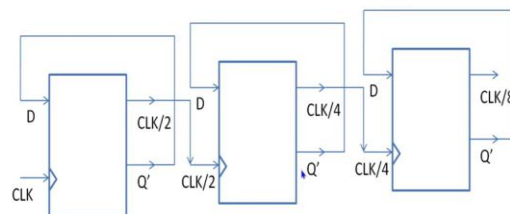
So, the duty cycle is independent of the previous clock, and is always 50 percent in this case, can we further divide it. So, can we divide it by any particular frequency? So, for that, I can repeat this this circuit? Yeah, so, one more thing that I can also design this clock divider by using T flip flop in case of T flip flop we know that if T is 0 then output remain as the previous state, but if T is 1 output keep on alternating.

So, instead of a D flip flop if I have a T flip flop in T flip flop I can always give this T as one then output will keep on fluctuating between one and 0 after alternate clock cycle. So, that means T flip flop can also be used for clock division. So, can we divide it by more than a value of 2? Yes.

(Refer Slide Time: 9:00)



More clock division



All clock signals are not synchronized!



So, for that, I can repeat this circuit. So, because of this one circuit, my output is clock by 2. Now, if I give this clock by 2 as an input to next D flip flop. So, I connect this Q to next clock and then the output is going to be clock by 4. So similarly, I can connect it further I can further connect it and then I can say that this output will go as an input clock for the another D flip flop so then this output signal is going to be clocked by 8.

So, this way we can keep on doing it recursively to essentially achieve at whatever division we would like to make. So, let us say I would like to divide clock by 10 then I would require 10 such flip flops, each of them each for each flip flop the circuit is going to be same or I can use all the T flip flops where T value is 1, but the output is connected as a clock to the next cascaded flip flop.

So, here output is connected to the clock of next cascaded flip flop. So, this could be interesting efficient design where the circuit is very minimal it just required a T flip flops and direct connections no other gate essentially. But there is one major bottleneck or major issue with this particular this particular clock divider circuit that all the clocks all the generated clocks. For example, this clock, this clock, this clock So, all of these clocks are not synchronized So, that means that all these clocks will not be going from low to high at same time.

And you remember that the reason that we are using clock is because we would like to have a synchronous circuit, because we would like to have synchronous circuits then all the clock transition should happen at same time, but here this clock and this clock would be delayed by a propagation delay of this D flip flop.

Similarly, this clock and this clock would have a timing difference of a propagation delay. So, because all of this become asynchronous clocks, so that may not be a very good idea to have a asynchronous clock. So, we will try to see some other mechanism to make a clock division which is synchronous in nature and which can be can still be efficient or we can still able to achieve a clock division using these flip flops, any of these flip flops D flip flop or JK flip flop, etc. So, this is how we can do clock division. Now, let us come to counting.

(Refer Slide Time: 12:09)

Counter



0000	1000
0001	1001
0010	1010
0011	1011
0100	1100
0101	1101
0110	1110
0111	1111

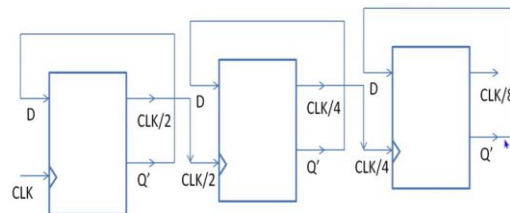
Frequency at every bit is half of other
Clock divider circuit can be used counter

Asynchronous counter

Digital Logic Design: Sequential Circuits.



More clock division



All clock signals are not synchronized!

Digital Logic Design: Sequential Circuits.



Now, what is accounting. So, basically let us say in one of the clock period first clock period the output is 0000, the next clock period output should be 001 then 2 then 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14 and 15. So, this counting could be anything and this counting is a very important digital circuit in the sense that it can help us to count number of clock cycles and as we have discussed in one of the previous lecture that this clock period define how much work can be done within clock period.

So, that is how we can we can have a good estimate of time also this clock, these counters can also work like a timers and they can help us in counting the time based on the counter the value of counter in that case could be huge. So, it could be 15 bit counter, 12 bit counter, 30 counter whatever. So, if you look closely at these values, what you will see that this is the

least significant bit, this least significant bit will keep on alternating every, after every number.

So, this become 0, this become 1 and similarly you can say this is 010101. So, this is every alternate number become every next number is the inverse of the first one. So, I can say this is again looking like a 1 bit counter, say 2 bit. Now, let us look at the second bit for basically 2s place bit, in case of 2s place I see 00 two times and then I see 11 two times then again 00 two times 11 two times so on so forth.

So, basically the frequency of 0 has got doubled. And similarly, frequency of one has got doubled and it is all looking like looking like a clock signal itself. Similarly, for the third bit, we see four 0s together and then four 1s together, then again, four 0s together, then again, four 1s together.

And for the fourth bit, we see that we have eight 0s together and then eight 1s together. So essentially, it looks like it looks like a clock signal. So, the other observation is yes frequency of every bit is half from the other. So, in this way we can use a clock divider circuit. So, this could be if I see my previous, if I see this circuit, so basically this could be clocked by 2 could be the first bit clock by 4 could be the second bit and clock by 8 could be the third bit and this way we can have another deep flip flop to have fourth bit.

So, my simple clock, my simple asynchronous clock divider can also work like a counter. Now, every time we are using the output of a D flip flop is a clock input to the next flip flop. But as I said earlier, this would be asynchronous counter because the counting values will not be synchronous with respect to clock my clock is changing in new flip flop in a different flip flop all the values are changing with respect to their own clock their own time.

So, yes but the message here is yes, we can use this clock divider also as a counter and the other in the other way around also that we can use counters as a clock divider circuit as well. So, the challenge with the asynchronous counter, both asynchronous counter as well as a asynchronous clock divider is that my clock is asynchronous my outputs are changing asynchronously.

So, how shall we design a synchronous counter or a synchronous clock divider circuit. So, if I want to do anything synchronous that means I am not supposed to connect any input to my clock, my clock should be connected to only the global clock. So, clock should not be modified at all.

(Refer Slide Time: 17:17)

Synchronous counter



- Clock will not be modified
 - One global clock to all FFs
- Next output depends on previous output/input



Digital Logic Design-Sequential Circuits

Now, how should I design a synchronous counter? So, for that, if I create a state table, I say that this is the present input then what would be the next input. So, based on that combination, then I can create a K map for all the next states and then we can create a counter.

(Refer Slide Time: 17:49)

3 bit synchronous counter



Q2	Q1	Q0	Q2 ⁺	Q1 ⁺	Q0 ⁺
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	1	0	1
1	0	1	1	1	0
1	1	0	1	1	1
1	1	1	0	0	0

$$Q0^+ = Q0'$$
$$Q1^+ = Q1'Q0 + Q1Q0'$$
$$Q2^+ = Q2'Q1Q0 + Q2(Q1' + Q0')$$



Digital Logic Design-Sequential Circuits

So, to illustrate this particular point, let us try to design a 3-bit synchronous counter. So, in 3-bit synchronous counter if my present state is 0 0 0 then the next state. So, what is the next state next state means after the positive edge of the clock, then the next state should become 0 0 1. Similarly, if the present state is 0 0 1, the next state after the clock edge should become 0

1 0. If the present state is 0 1 0 the next state should become 0 1 1 and then 1 0 0, then 1 0 1, 1 1 0, 1 1 1.

And after the input is 1 1 1, it should only go back to 0 0 0. Because it is a 3-bit counter. So, after 7 it has to come back to 0. So, how do we solve this, now what should we go, what should we do ahead next thing. So, we know that whatever is the, whatever we would like to have as the output if we give that as an input to my D in a D flip flop, if I give that as input to D, then that will become that would be transported or that would be given as the output in the next state.

So, whenever the clock positive edge or the clock will come that D would become Q plus Q0 plus. Similarly, for these flip flops, I would require 3 flip flops again, I would require 3 D flip flops, for each flip flop, this Q0 plus value I can say this is actually the D value or the input value of that particular flip flop.

So similarly, whatever I would like to have the next state, if that is the actually the input of that particular flip flop. So, we would like to create to know what are the logic expression for Q0 plus Q1 plus and Q2 plus and then that we know already that we need to solve over K map for these 3 inputs Q0, Q1 and Q2.

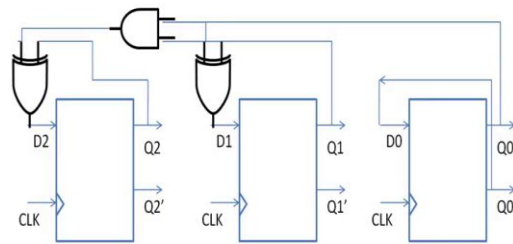
So, here I have solved this K map directly, then you can work out yourself or maybe you can quickly see that this Q0 plus. So, every time Q0 is 0 then it becomes 1 this at whenever this is 1 it will become 0 irrespective of Q1 and Q2. So, that means Q0 plus is equal to Q0 dash. Similarly, Q1 plus it is 1 only in these two circumstances that means, Q1 dash Q0 or Q 1 Q0 dash, which is also an XOR operation.

So, here also this is 1 but the condition is similar that it is Q1 dash Q0 or Q1 Q0 dash, for Q2 plus we see it is 1, for these four cases the first case is Q2 dash and Q1 Q0 or it is Q2 and Q1 dash plus Q0 dash. So, this whole could be combined as Q1 dash plus Q0 dash. So, if I would like to design a synchronous counter then I as I said so, this should be the input of this this D flip flops.

(Refer Slide Time: 21:36)



3 Bit counter



$$D0 = Q0'$$

$$D1 = Q1'Q0 + Q1Q0'$$

$$D2 = Q2'Q1Q0 + Q2(Q1' + Q0')$$

$$= Q2 \oplus Q1Q0$$

Digital Logic Design-Sequential Circuits.



Modulo 5 bit counter

Q2	Q1	Q0	Q2'	Q1'	Q0'
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	1	0	1
1	0	1	0	0	0
1	1	0	x	x	X
1	1	1	x	x	X

$$Q0+ = Q0'$$

$$Q1+ = Q2Q1'Q0 + Q1Q0'$$

$$Q2+ = Q1Q0 + Q2Q0'$$

Clock division by 6

Read This as Modulo 6 counter



So, I can say that D0 would be equal to Q0 dash D1 is equal to Q1 XOR Q0 and D2 I can simplify this equation as Q2 XOR Q1 Q0 and again implement this using these 3D flip flops where D0 is equal to Q0 dash and D1 is equal to Q1 and Q0 is XOR and D2 is equal to XOR of Q2 and AND of Q1 and Q0.

So, this implementation can help us in doing a synchronous 3-bit counter. So, as you say it is synchronous, that essentially means that all the values of Q0, Q1 and Q2 they would be available at the same time equal to propagation delay of these D flip flop. So, after a clock period so whenever the active edge of a clock, let us say it is a positive activate, so as soon as positive active edge or the clock will come after propagation delay Q0, Q1 and Q2 would be available as the output so they are called synchronous outputs or synchronous counters.

Now, let us say this was a 3-bit flip flop, which was counting from 0, 1, 2, 3, 4, 5, 6, 7. Let us say I do not want to count from 0 to 7 but I just wanted to count from this, let us say only 0 to 5; 0, 1, 2, 3, 4, 5. And whenever 5 will come I will go back to 0 0 0. So, what would happen if my input is 1 1 0, 1 1 1 both of these inputs are invalid because if counter will start at 0, these values will never come.

So, as soon as 0 0 0 0, 1 2 3 4 and 5, from 5 it will go back to 0. So, that means these input combinations will never ever come. So, the same technique, same method we can use, same technique and same method we can use to design these 5 bits modulo counter this is called modulo because it again after 5 it again go back to 0. So, similarly there could be if it is a modulo 12 counter, so that means after 12 it will again go back to 0, so I should have said modulo 5 counter the bit is not required, so it is a modulo 5 counter. Now, to solve this again we have to write K maps and K map for Q0 plus, Q1 plus and Q2 plus.

Now, we can try to take advantage of these don't care condition wherever it is possible. And then we can see that Q0 plus would again be equal to Q0 dash and Q1 plus. So, this Q1 plus these are the two values of where Q1 is plus. So Q1, Q1 plus is 1. So, one of the value would be happening when Q2 is 0, Q1 is 0 and Q0 is 1, the other we can combine with a don't care condition. So, it will become Q1 and Q0. Similarly, for Q2 dash plus again we can create a truth table and so, sorry truth table is there. So, we can create a K map and then can optimize it.

So, in this case, I have solved it become Q1 Q0 plus Q2 Q0 dash. So, this is how we can design any modulo counter. So, you see one more thing here that this modulo 5 counter can also be used to design divide clock division by 6 actually, it is a clock division by 6. So, in clock division by 6 you can see that this output will keep on repeating itself.

So, you will see that after it will go 0 1 2 3 4 5. So, when the output is going to be 5, so, this output is will keep on repeating itself that means, if I take Q2 as my clock divisions, then I can see that my Q2 is actually a clock divided by 6 because four times it will then return 0 and two times it will return 1.

So, which means that duty cycle is going to be a one third. So, for a one third of the time it is going to be 1 and for two third of the time it is going to be 0 and but because this is going to repeat again and again this could be a clock division by 6. So, in this way if we use these modulo counters, these modular counters can help us in dividing the clock by any particular number it need not to be 2 is to power number.

So, let us say if I want to design a clock divider by 1000 so I can design it accordingly. So, I can although the duty cycle will not be 50 percent but we would be able to design a clock divider for by any particular number. Yes, so this may also be worth saying here that clock division is easier to be done using digital circuits essentially using a flip flop, but clock multiplication, let us say the input clock period is 100 megahertz.



So, there is no method, no clear method or no systematic method that can help us in achieving a clock which is of 200 megahertz or clock which is faster. So, from 100 megahertz, we can go down to a slower clock, but faster clock will not be possible. And in creating a slower clock, we can be we can calibrate or we can basically do any we can divide by any particular number of our choice.

(Refer Slide Time: 28:44)

Sequence generator

- Any sequence can be generated
- Steps
 - Number of bits in sequence decide the number of flip-flop
 - Next state equations by writing the truth table
 - Find Boolean expression for each next state bit
- Example Homework
 - 0 -> 2 -> 1 -> 4 -> 6 -> 3 -> 0

Digital Logic Design: Sequential Circuits



So now, one last application related application here could be we can we can essentially use this particular method to generate any sequence. Any sequence means that, let us say I would like to have the example I am taking. So, for example, initially the value is 0 then I would like to in the next clock cycle, I want to make it 2 and then 1 then 4, then 6 then 3 then 0. So, and then this can keep on repeating.

So, similarly, any sequence of numbers if I would like to generate, then that could be generated that whatever is the next number, I would like it to be in the next clock cycle according to that I can write my next state equations. And based on those state equations, we can find out what would be the we can we can design our logic so that the next number would be equal to whatever sequence we would like to write.

What would be the steps here, what would be the method. So basically, let us say first of all, we need to find out how many bits would be there. So, for example, in this case, it would require 3 bits, but any sequence of numbers, then whatever is the maximum number \log_2 of that particular maximum number can tell us the number of bits in that sequence generator, whatever is the number of bits that many number of flip flops, memory elements we require.

Now, after that we can write a truth table; in the truth table, we can write let us say the present state is 0 then we can certainly say that next state is 2, if the present state is 2 next state is 1. Similarly, the present state is 1 the next state is 4. So, for all the transitions we write the present state in the next state and for each next state bit.

So here for example, there would be 3 bits for each next state bit we can write a Boolean expression and that Boolean expression could be assigned to input D, D input of that particular flip flop. So and D would be transferred to Q at the positive edge of the of the clock so that means this sequence generator, any sequence generator can be designed using D flip flop.

(Refer Slide Time: 31:18)

Summary



- D Flip-flop based design
 - Any other flip-flop
 - Clock divider
 - Counter
 - Modulo counters
 - Sequence generators

Digital Logic Design: Sequential Circuits.



So, with this I would like to close this lecture and in summary, what we have seen in this lecture that we can use these D flip flops to design any other flip flop like JK flip flop or T flip flop etcetera. And we have also seen how these D flip flops can be used to design a clock divider, we have seen both clock divider which is synchronous and another clock divider which is asynchronous.

Similarly, we can use these D flip flop to design counters also, counters could be synchronous or asynchronous. In case of asynchronous counters, the output of one D flip flop is given as the input, clock input of the next flip flop, but in case of a synchronous counters we do not modify the clock because the clock is my synchronizing signal. So, and value of D is calculated based on a truth table.

So, that is how we can design synchronous counters. And using the similar technique the way we have designed asynchronous counter we can also design modulo counters we can also design any kind of a sequence generator where sequence is some integer number and from one integer number we would like to go to another integer number. So, that kind of a sequence generator can be can be designed using the flip flop. So, with this I would close, thank you very much.