

**Digital Circuits**  
**Prof. Santanu Chattopadhyay**  
**Department of Electronics and Electrical Communication Engineering**  
**Indian Institute of Technology, Kharagpur**

**Lecture - 33**  
**Sequential Circuits (Contd.)**

So, next application that we look into is the transmission of data in serial format.

(Refer Slide Time: 00:18)

**Transmission of data — serial format**

Data often has to be transmitted from one computer to another, or from a computer to peripheral equipment (printer, modem, ...). This can be done in:

- **serial format**, one bit at a time
- **parallel format**, several bits at a time (e.g. byte at a time, 8 bits)

Serial format is most commonly used because it is simpler. Only a few wires are needed:

- traditional **serial 'COM' ports (RS-232)** need only 3 wires (transmitted data, received data and ground — but more may be used for control)
- **universal serial bus (USB, common on modern computers)** uses 4 wires (two for differential data plus power and ground)

Traditional serial transmission was slow but modern systems use much faster rates (USB version 1 up to 12 Mbits per second, FireWire 1 up to 400 Mbits per second), version 2 of both even faster.

simple serial bit stream

1 0 0 1 1 0 1 1 0 1 0 1

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Now, as we know that the data it has to be transmitted from one computer to another computer or one system to another system and normally what you have may, may be one computer to your printer or modem so like that.

Now while we are doing this transmission, so, it can be done in two formats either in serial format or in parallel format. As I already said that the advantage of serial transmission is that, the number of lines that you need to draw, so that is less ok. So, you have to take only if you have single line for carrying 1 bit value 1 bit at a time is transmitted, but when you are doing in parallel format, then several bits will be transmitted serially, so if you are transmitted parallelly.

So, if you are transmitting say 8 bits parallelly, then you have to have 8 parallel lines running between the source and destination. So, this is that is why the serial transmission is quite popular in many applications. So, this is simpler and only point few wires are

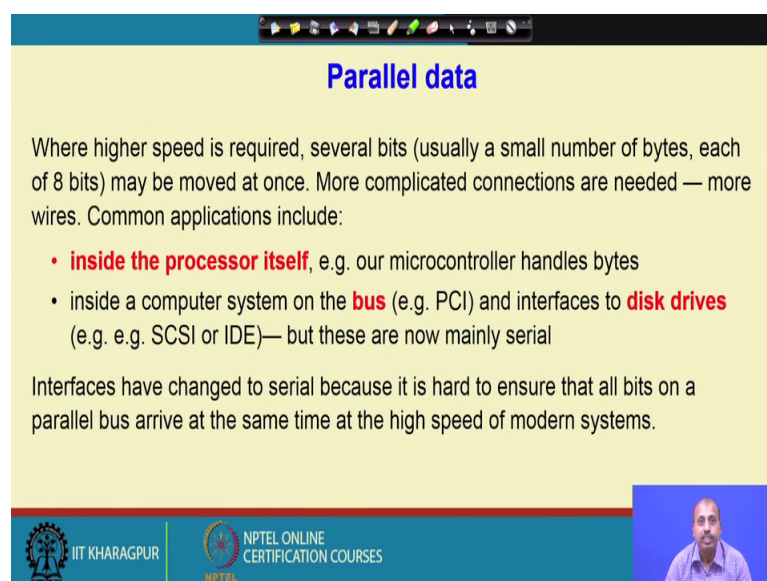
needed. So, traditionally if you look into the computer the see the serial com ports RS 232 ports. So, they need only 3 wires transmitted data, received data and ground; but there there some more controls are needed which is not stated here for doing the set ups and all, so that is there.

Another serial communication is universal serial bus or USB standard that uses 4 wires two for differential data, plus power and ground. So, that way this both of them are serial. So, in case of COM port we need one transmit data, one receive data and ground and in case of USB we need only 4 wires of course, for serial for this COM port RS 232 communication.

So, more number of a lines are needed, but for USB it is 4 only. So, that way USB has become a better standard, but anyway we are looking into a serial transmission. So, traditional serial transmission was slow, but modern system are faster like USB version 1 is up to 12 megabits per second, Firewire up to 1 up to Firewire 1 is up to 400 megabits per second version 2 is even faster. So, we now we have got even USB version 3.0 which is still faster.

So, that way, but essentially what is happening is that for serial transmission. So, we transmit in transmit them as a bit pattern. So, this so 1 is transmitted as logic high 0 is transmitted as logic low so, that way this bit pattern is transmitted serially.

(Refer Slide Time: 02:54)




**Parallel data**

Where higher speed is required, several bits (usually a small number of bytes, each of 8 bits) may be moved at once. More complicated connections are needed — more wires. Common applications include:

- **inside the processor itself**, e.g. our microcontroller handles bytes
- inside a computer system on the **bus** (e.g. PCI) and interfaces to **disk drives** (e.g. e.g. SCSI or IDE)— but these are now mainly serial

Interfaces have changed to serial because it is hard to ensure that all bits on a parallel bus arrive at the same time at the high speed of modern systems.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES



Whereas when higher speed data transfer is required, we need to transfer several bits in parallelly and usually a small number of bytes each of say 8 bits that may be moved at a time, more complicated connections are needed more wires and. So, some applications they will need it one difficulty with the serial transmission is that since you are transmitting serially. So, time needed for the total transmission is higher compared to a parallel transmission.

So, it is the parallel transmission takes place inside the processor itself for example, microcontrollers, microprocessors they can handle in terms of bytes. Inside a computer system on the bus for example, you are connecting this PCI you have you are having PCI bus that connects the processor to other disk drives and all or SCSI, IDE disk drives. And they are all having this parallel communication, but because they are they were initially parallel, because of this thing that this transmission will be faster, but now the serial communication is also being used.

Now, interfacing that you have, so from the processor it comes out as a parallel data, but the transmission is serial. So, we need to have a parallel to serial converter. So, this interfaces are changed to serial because it is hard to ensure that all bits on a parallel bus arrive at the same time at high speed of modern systems. So, you are you are expecting that an 8 bit data, but that 8 bit data if it is coming through 8 bit parallel lines, then it is there is a possibility that this 8 parallel bits will not arrive at the same point of time. So, there may be a synchronization problem between this 8 bits.

So, it is better that we go for serial communication was this transmission will be done serially. And also there are other effects like cross talk between the parallel lines which effects the logic value that you are sending over the bits.

(Refer Slide Time: 04:55)

**Parallel data**

Where higher speed is required, several bits (usually a small number of bytes, each of 8 bits) may be moved at once. More complicated connections are needed — more wires. Common applications include:

- **inside the processor itself**, e.g. our microcontroller handles bytes
- inside a computer system on the **bus** (e.g. PCI) and interfaces to **disk drives** (e.g. e.g. SCSI or IDE)— but these are now mainly serial

Interfaces have changed to serial because it is hard to ensure that all bits on a parallel bus arrive at the same time at the high speed of modern systems.

**How do you interface a serial device to a computer?**

How do we interface an external device that transmits serially with the bus of a computer that transfers one byte (8 bits) at a time?

- **Use a shift register.**

In practice this would almost certainly be buried inside a larger circuit called a **UART** (universal asynchronous receiver transmitter) or something similar.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Where higher speed is necessary, so we have several bits are transmitted that we moved at once, and more complicated connections are needed more wires like inside the processor itself or inside the computer. So, how do you change this serial device to a compute to a computer? So, how do we interface an external device that transmits serially with the bus of a computer that transfer only 1 bit.

So, this is this transmission has to be via some shift register, because then shift register only it you can do a parallel to serial conversion and we have got, we have already seen that they can use some sort of UART for doing this communication.

(Refer Slide Time: 05:33)

**Use of shift register to serialize data**

parallel data in

parallel load

serial output

Extra logic is added to the basic shift register so that all the flip-flops can be loaded in **parallel** (simultaneously), controlled by a shift/load input.

Once the data have been loaded, the clock is enabled and the values are shifted once per clock cycle. This causes the input data to be transferred to the output, one bit at a time — **serial output** (PISO).

The opposite process is used to read in serial data, fill up the shift register, and transfer it in parallel to a bus when the register is full (SIPO).

The register can also be parallel input – parallel output (PIPO).

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

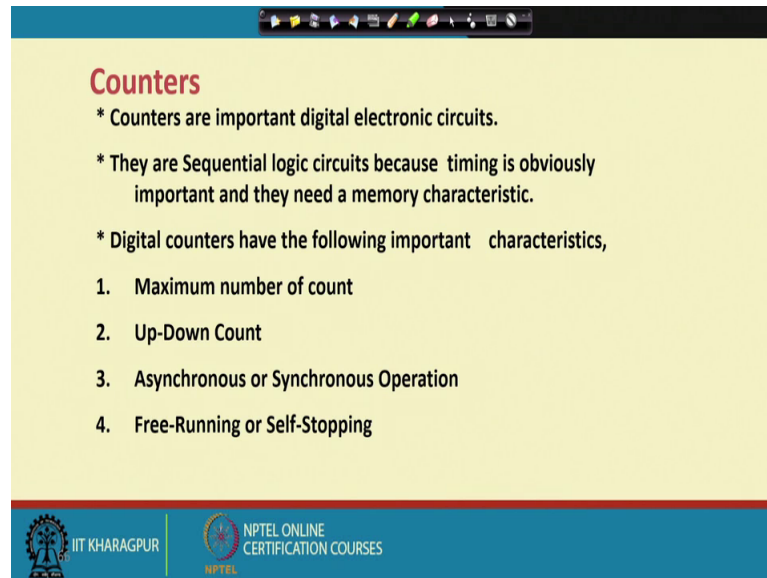
So, this diagram actually explains how can we have this parallel to serial conversion. So, parallel data is loaded parallelly on to these D flip flops and then this D flip flops are otherwise connected in a series fashion. So, this logic we have seen previously parallel in serial out type of register configuration or this is known as PISO structure, parallel input serial output type of structure.

So, you can also at the receiving end, so we can have this SIPO structure serial input parallel output type of structure. Where the serially the bits will be coming and then they will be converted into parallel data and given to the processor or we can also have parallel input parallel output type of connection.

Next will be looking into another very important digital circuit component which is known as counter so, counters are normally used to count some event ok. So, suppose we are counting like how much time, time is lapsed from the beginning of certain event or when the items are moving on a conveyor belts. So, you want to keep a count on how many items have passed through the conveyor belt. So, like that we can have different application where we need to really count the number of occurrences of the events.

So, they are so how do you do it like occurrences of an event may be coming as a trigger to your circuit, but the circuit has to memorize like how many such things it has seen ok.

(Refer Slide Time: 07:03)



**Counters**

- \* Counters are important digital electronic circuits.
- \* They are Sequential logic circuits because timing is obviously important and they need a memory characteristic.
- \* Digital counters have the following important characteristics,
  1. Maximum number of count
  2. Up-Down Count
  3. Asynchronous or Synchronous Operation
  4. Free-Running or Self-Stopping

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

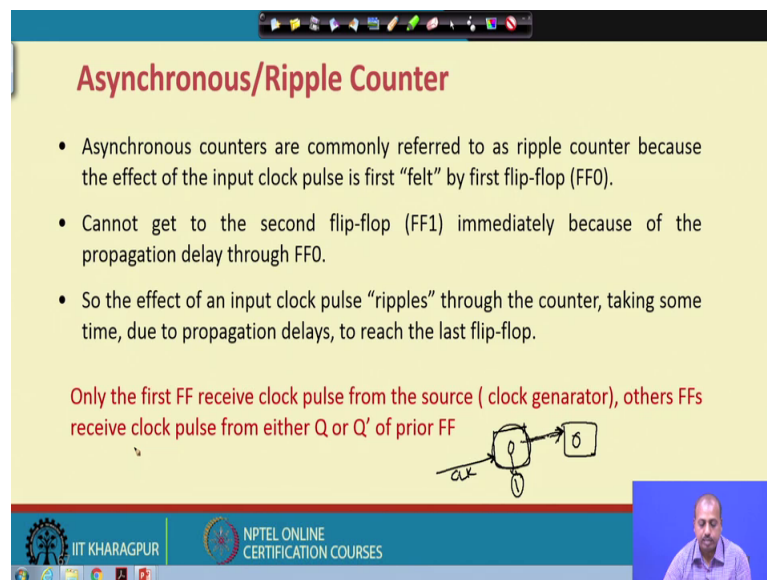
So, this will be done by means of counters, so counters are important digital electronic circuits they are sequential logic circuits because timing is obviously, important. And they need a memory to need a need a memory characteristic because after every it has to after every event it has to remember, how many events it has seen? So, that is why it is sequential one in nature and digital counters have the following characteristics like there, there will be a maximum number of count because ultimately the values are stored in some flip flops. So, depending upon the number of flip flops that employ to hold the count value there will be a limit.

Like for example, if I keep say 4 flip flops to keep the count value, then after the values becomes all one then at the next at the next time instant when I try to increment this 1 1 1 1 by 1, so it becomes all 0. So, there is a maximum count value which is 15, in this case the counter may be an up counter or a down counter. So, we can we can start from some initial value and the counter may be instructed to increase its value by one, after each event occurrence or it may be that I start with some initial value and then with the occurrence of an event we start down counting.

So, this up down counting may be there, the counters may be designed in both asynchronous and synchronous fashion. So, will see that and we may have some free running counter or self-stopping counter. So, may be as soon as we power on the system the counter. So, it may be it may start running continually without any, any intervention

from outside or may be self-stopping after the counter has reached a particular final value it stops at that point. So, depending upon the type of application that we have we have to choose a particular type of counter, so in our course. So, will be looking into how to design this different types of counters.

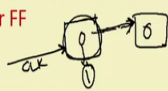
(Refer Slide Time: 09:02)



**Asynchronous/Ripple Counter**

- Asynchronous counters are commonly referred to as ripple counter because the effect of the input clock pulse is first “felt” by first flip-flop (FF0).
- Cannot get to the second flip-flop (FF1) immediately because of the propagation delay through FF0.
- So the effect of an input clock pulse “ripples” through the counter, taking some time, due to propagation delays, to reach the last flip-flop.

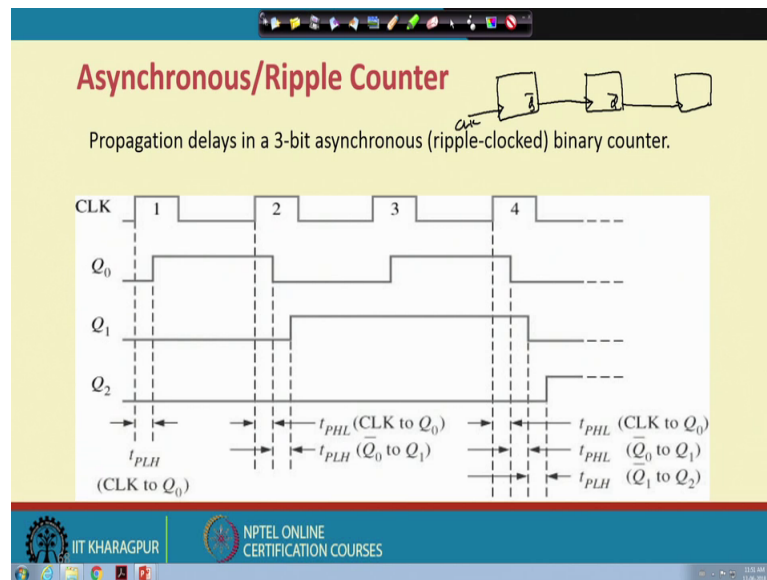
Only the first FF receive clock pulse from the source ( clock generator), others FFs receive clock pulse from either Q or Q' of prior FF



The slide features a yellow background with a red title. It contains three bullet points explaining the ripple counter's operation. Below the text, a red line of text states that only the first flip-flop receives the clock pulse from the source, while others receive it from the output of the previous flip-flop. A hand-drawn diagram shows a clock pulse 'ax' entering the first flip-flop (FF0), which is connected to the second flip-flop (FF1). The slide also includes logos for IIT KHARAGPUR and NPTEL ONLINE CERTIFICATION COURSES, and a small video inset of a presenter.

The first counter that will look into is known as asynchronous ripple counter. So, asynchronous counters are commonly referred to as ripple counter, because the effect of the input clock pulse is first felt by the first flip flop. And then it cannot go get to the second flip flop immediately because of the propagation delay of the first flip flop.

(Refer Slide Time: 09:24)



So, it is like this so if I have got say if I takes a 2 flip flops. So, this is 1 flip flop and this is another flip flop then. So, if so this, the clock signal that is coming to the first flip flop then. So, the value when this when this clock signal comes when the first clock suppose I have initially made both of them equal to 0. When both of them are equal to 0 now the first when the clock pulse comes, so this flip flop makes a transition to one, but as an input this flip flop sees still the output of the previous flip flop ok. At that way so it will not see this one immediately when the clock pulse will be arriving.

So, it will still see it as 0 and this (Refer Time: 10:13) this flip flop has got a propagation delay that is why the second flip flop will not see the, it will not get the it will not get the one value directly there, so it will not there. So, as a result the effect of the input clock pulse ripples through the counter and so will see how does it operate only the first flip flop.

So, in the structure that we have follow only the first flip flop receive clock pulse from the source other flip flops receive clock pulses from either Q or Q bar of the previous flip flop. So, what do we mean is that, if I have got say three such flip flops forming the counter then one possible structure may be like this.



(Refer Slide Time: 10:53)

**Asynchronous/Ripple Counter**

- Asynchronous counters are commonly referred to as ripple counter because the effect of the input clock pulse is first “felt” by first flip-flop (FF0).
- Cannot get to the second flip-flop (FF1) immediately because of the propagation delay through FF0.
- So the effect of an input clock pulse “ripples” through the counter, taking some time, due to propagation delays, to reach the last flip-flop.

Only the first FF receive clock pulse from the source ( clock generator), others FFs receive clock pulse from either Q or Q' of prior FF

The clock pulse is connected to the first flip flop, and the second flip flop do the clock pulse is connected from the Q output of the second flip flop. Similarly, for the third flip flop the clock pulse may be connected from the Q output of the second flip flop. As a result the clock pulse that, so this clock is coming from the outside world. So, when the clock pulse comes it only the first flip flop sees the clock rest of the flip flops will not see. When this Q makes it transition from say 0 to 1 or 1 to 0 then only this is being felt by the second flip flop.

Similarly, when this flip flop makes a transition from 1 to 0 or 0 to 1 then only this clock will be felt by the third flip flop. So, that is why the last statement that we have here that this only the first flip flop receive clock pulse from the source or the clock generator and other flip flops they receive clock pulse from either Q or Q bar of the previous flip flop. So, if you have got these type of connection then you see suppose, suppose this is the free running clock that we have ok. So, clock is high for this much amount of time then it is low for this much time again it is like this, so it goes like this.

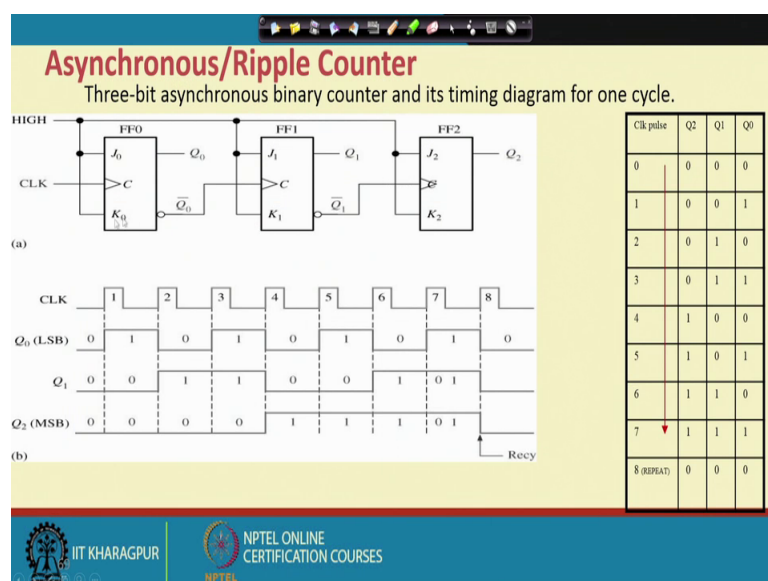
So, per the behavior that we have got so, if we have if we connect this clock, clock to the first flip flop, then due the delay of the first flip flop the first flip flop will initially the value was 0. So, it will it will become one after sometime and that time is given by this t<sub>PLH</sub>, this is the propagation time from clock to Q Q 0 of the first flip flop, so it makes a transition here.

For the second flip flop, so it does not see the clock pulse immediately, but what happens is that it will see the clock pulse when this thing when this clock pulse to be high when I have got this Q 0 output to be equal to high, so it will see this here. So, then when this is when this particular Q 0 is going low. So, it will see that there is a transition in the Q 0 line. So, then this fellow will be making a transition getting Q 1 to be high and after sometime those this Q 2, Q 2 will become high. So, at this point, so this is the Q 1 is making at transition from high to low. So, now, it will be making a transition like this.

So, here I have connected, so in this particular case it is connected that it is assume that clock is connected to Q 0. So, the counter structure that is assumed here is like this. So, this is the first flip flop this is the second flip flop and this is the third flip flop. So, this is the clock is connected to the first flip flop like this and then this the second line second flip flop. So, we have got this Q bar output connected to the clock and here also the Q bar output is connected to the clock. So, you see that at this time this Q naught is making a transition from high to low that means, that this Q, Q naught bar is making a transition from low to high, so that is actually sensed here.

Similarly, at this point Q 1 makes a transition from high to low. So, this Q 1 bar is making a transition from low to high and that is sensed by the third flip flop at this point. So, that way we can have this connection of this 3 bit counter to get this rippling effect.

(Refer Slide Time: 14:37)



So, this is the structure, so what we what has been done. So, all the 3 flip flops we have we have taken J K type of flip flop and this J, J and k, so they are tied high. So, you remember that in a J K flip flop if J and K both are tied high then whenever the clock comes so it will make a transition. Now you see that if this is the free running clock signal that we have so this Q 0 ok.

So, this particular flip flop, so this will transit at every rising edge of the clock. So, initially if the flip flops are content all 0 then at the first clock edge. So, this Q 0 makes a transition it becomes 1 then at the next clock edge, so Q 0 will become 0. So, it will go like this third clock edge again it will become high so it will go, so Q 0 timing diagram is like this

Now, once we have got the Q 0 time timing diagram Q 0 bars timing diagram is just the reverse of this. So, whenever Q 0 makes a transition from 1 to 0 Q 0 bar makes a transition from 0 to 1 and that is seen as the clock signal for flip flop 1. So, this Q 1 makes a transition when Q 0 goes from 1 to 0 or equivalently Q 0 bar goes from 0 to 1, so it makes a transition here.

So, next transition of Q 1 occurs at this point ok. So, at this point again Q 1 makes a transition. So, you see now if you if you look into this count value if you say that this Q 2 is the most significant bit of my count, and Q 0 is the least significant bit of my count. So, initially if all this flip flops are assumed to be their content is assumed to be 0. So, initially it is 0 0 0 after the first clock pulse high, so it becomes 0 0 1, after the second clock pulse it becomes 0 1 0. Then it is 0 1 1 then 1 0 0 1 0 1 1 1 0 1 1 1 and then there is a mistake this two zeros should be here, so it becomes 0 0 0.

Now, from this point onwards this is basically recycling now it again generates the same sequence. So, we can say that if we try to look into the behavior of this counter. So, it start with 0 0 0 then at successive clock pulses at clock pulse 1 the count value will be 0 0 1 at clock pulse 2 0 1 0 clock pulse clock pulse 3 0 1 1. So, it will go like this till it comes to this step 8 where it is 0 0 0, and now it goes on like that.

So, it is a 3 bit asynchronous binary counter we have got the timing diagram for one clock cycle that is for one cycle that is for 0 to 8. So, that is since it is a 3 bit counter. So, after 8 it will be over flowing and it will be coming back to 0 after 7 basically after 7 it will coming back to 0 and now this sequence will be repeating.

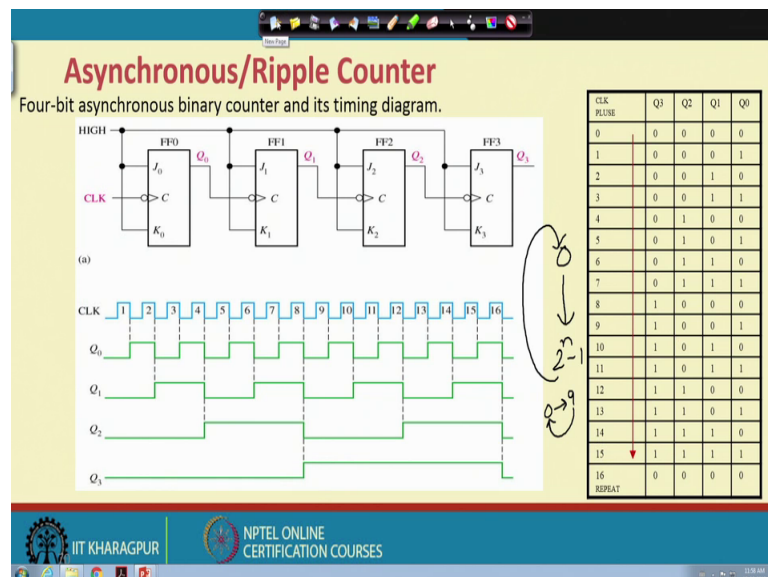
Now, you see that this clock signal if you take it as an as some event for example, it may be on a conveyor belt items are passing, and there is a sensor light sensor which senses that whether some item is passing or not and whenever some item blocks that light. So, we have we suppose we have got a 1 ok, so that way.

So, this clock signal is generated from that items moving on to the conveyor belt and accordingly this counter will now be counting like how many times it is getting blocked or if we in intro if we have this circuit in the in installed at the door of a room. And we have got appropriate sensors for sensing whether somebody is entering into the room or not.

So, every every time a person enters or some obstruction occurs then the light may be there is a led and a sensor. And they will be sensing that the light is getting obstructed accordingly that can be treated as an event and every time that event occurs it will generate a pulse like this and accordingly the count value will get incremented at the end of the day, we can have a count about how many persons entered the room on that particular day.

So, this way we can have this ripple counter implemented and used in digital systems.

(Refer Slide Time: 18:51)



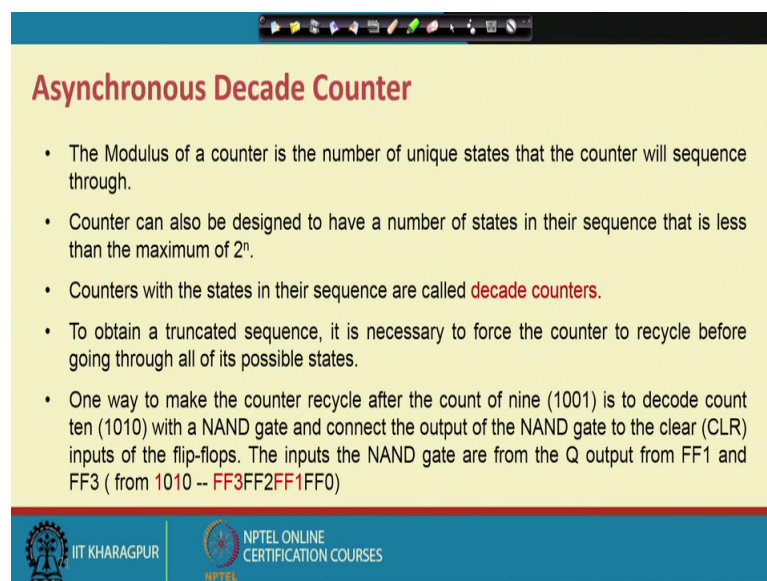
We can also have asynchronous ripple counter, so this is in this case 4 bit this is a 4 bit asynchronous ripple counter. So, previously it was a 3 bit counter now we have got a 4

bit counter. Where this  $Q_0$  is connected to the clock of the  $Q_0$  bar basically this  $Q_0$  and this is inversion. So that means, the falling edge of the clock the transitions will take place.

So, you can reason out in a similar fashion that if you start at time 0 then initially all this contents are 0. Then if you give some clock pulse at the following edge of the clock this  $Q_0$  will become 1. Similarly, this flip flop will transit at the falling edges of  $Q_0$ . So, at the falling edges of  $Q_0$  this  $Q_1$  makes a transition at falling edges of  $Q_1$   $Q_2$  makes transitions, so like this and at falling edge of  $Q_2$   $Q_3$  makes a transition.

So, here also if you look into the pattern, so initiate 0 0 0 0 then at this time it is 0 0 0 1 then it is 0 0 1 0, so it generates this particular sequence. So, 0 0 0 0 to 1 1 1 1 and then it repeats the sequence from the state 16. So, this way we have this 4 bit asynchronous binary counter.

(Refer Slide Time: 20:12)



**Asynchronous Decade Counter**

- The Modulus of a counter is the number of unique states that the counter will sequence through.
- Counter can also be designed to have a number of states in their sequence that is less than the maximum of  $2^n$ .
- Counters with the states in their sequence are called **decade counters**.
- To obtain a truncated sequence, it is necessary to force the counter to recycle before going through all of its possible states.
- One way to make the counter recycle after the count of nine (1001) is to decode count ten (1010) with a NAND gate and connect the output of the NAND gate to the clear (CLR) inputs of the flip-flops. The inputs the NAND gate are from the  $Q$  output from FF1 and FF3 ( from 1010 -- FF3FF2FF1FF0)

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Sometimes, we don't we don't want to count up to say the maximum capacity. So, if you look into this diagram, so if I put  $n$  such flip flops then you can understand that the count value, if I start at 0 it will go up to  $2$  to the power  $n$  minus 1 and then it will repeat, so this is the standard behavior.

Now, many times we don't want to go up to  $2$  to the power  $n$  for example, in a 4 bit counter we may be interested to count up to from 0 to 9 only, and after 9 we want to

come back to 0. So, that is if we are following some decimal system may be may be interested in some counting up to 0 to 9.

So, this type of counters they are known as decade counter that is it is modulo 10 counter ok. So, for getting that type of counter, so we can do some modification to the circuit that we have drawn here. So, modulus of a counter is the number of unique states that the counter will sequence through. Like previously that was counter that we have so that was mod 16 counter because it was going through 0 to 15 16 distinct states distinct unique states.

The counter can also be designed to have a number of states in their sequence that is less than the maximum value of 2 to the power n. So, decade counter is an example of that category, so where we have got up to 10 ok. So, how can we do that so one possible way of designing a decade counter is that after the counter is, we want to recycle the counter after 9. So, for that as soon as the count value becomes 10 10, maybe we can put some NAND gate and then make connect it to the clear line.

(Refer Slide Time: 21:58)

**Asynchronous Decade Counter**  
An asynchronously clocked decade counter with asynchronous recycling.

CLK PULSE	Q3	Q2	Q1	Q0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10 GLITCH	0	0	0	0
11	0	0	0	1
12	0	0	1	0
13	0	0	1	1
14	0	1	0	0
15	0	1	0	1
16	0	1	1	0

mod-10

IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES

So, this diagram actually shows it. So, when the count value becomes 0 sorry count value becomes 9, after that in the next clock pulse the count value becomes 10 that is 1 0 1 0. So, when it is 1 0 1 0, so the combination is sorry. So, this is the most significant bit so this is 1 0 1 0. So, when it comes like this now what we are doing we are taking out, so this one here and this one here.

So, these two are taken, so they are given to this NAND gate and so NAND gate output becomes 0. And they are connected to the clear input of this flip flops and since this is clear bar connection. So, as soon as this clear line becomes 0. So, all this flip flops will become 0, so their content will become 0 they will become 0.

So, essential what happens is that after this counting 9 when it goes to 10, it immediately comes back to 0 of course, for that is small amount of time it will show the value 10 that is determined by the delay of this NAND gate and the delay of this clear in circuitry in the individual flip flops, but after that the value will become equal to 0. And if that value is much less compared to the clock duration then that value may be may be negligible, but you will a glitch like this. So, here you will get a glitch momentarily the value will become 1 0 1 0 and then value will be momentarily become 1 0 1 0 and then it will be going to all 0 ok.

So, this way you can design you can put some extra gate. So, whatever count value whatever mod value is given suppose, I am asked to design a mod 11 counter, mod 11 counter means 11 in that case, so if it is mod 11. So, it is 1 0 1 1. So, in that case what I have to do is that I have to take a NAND gate where this connections will be there. So, this is 1 0 1 1, so these two ones are already there, so I have to take this one also and connect it as another input to this NAND gate.

(Refer Slide Time: 24:21)

### Asynchronous Decade Counter

An asynchronously clocked decade counter with asynchronous recycling.

(a)

CLK: 1 2 3 4 5 6 7 8 9 10

Q<sub>0</sub>: [Timing diagram showing Q0 transitions]

Q<sub>1</sub>: [Timing diagram showing Q1 transitions]

Q<sub>2</sub>: [Timing diagram showing Q2 transitions]

Q<sub>3</sub>: [Timing diagram showing Q3 transitions]

CLR: [Timing diagram showing CLR pulses at clock 10]

Glitch →

CLK PULSE	Q <sub>3</sub>	Q <sub>2</sub>	Q <sub>1</sub>	Q <sub>0</sub>
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10 GLITCH	0	0	0	0
11	0	0	0	1
12	0	0	1	0
13	0	0	1	1
14	0	1	0	0
15	0	1	0	1
16	0	1	1	0

So, then what will happen is that when the count value becomes 1 1 0 1 or 1 0 1 1 then this clear value will come and it will reset the counter. So, that way you can very easily decide this resetting value. So, depending upon the mod value that you have so, you can decide whenever that particular count is reached. So, you can say that I will stop at that point and you can get this the decayed counter or whatever be the mod value. So, accordingly you can design the counter.

(Refer Slide Time: 24:59)

**Synchronous binary Counter**

The term Synchronous refers to events that have a fixed time relationship with each other AND receive clock pulse from a common source

**2-bit synchronous binary counter.**

The diagram shows a circuit with two J-K flip-flops, FF0 and FF1. The J and K inputs of FF0 are tied to HIGH. The clock (CLK) is connected to both flip-flops. The output of FF0 (Q0) is connected to the J and K inputs of FF1. The timing diagram shows CLK, Q0, and Q1 signals over four clock cycles. Q0 toggles on every rising edge of CLK, and Q1 toggles on every rising edge of Q0.

So, you can also design counters in a synchronous fashion. So, previously we have seen the counters which asynchronous in nature because the individual flip flop they were not operating at the same clock. So, all the flip flops they had clocks, but the clock was not common throughout the design; but.

So, another possible type of design is when the same clock signal is fed to all the flip flop. So, that will give rise to synchronous design and this synchronous counter design is also simple, like in this particular case what we do? It is a 2 bit synchronous counter. So, there is a this clock signal is connected directly to both the flip flops, but this is the J K line. So, this J and K lines are tied high for the first flip flop 0, and for the second line J and K, so they are tied high together and connected to the Q 0 lines.

So, whenever this Q 0 equal to 1 then only this J 1 and K 1 they get the value 1. So, as result this flip flop will toggle. So, if you look into the timing diagram, you see that the first flip flop Q 0. So, it toggles after at the rising edge of every clock ok, because this J



and K are tied high. So, they are actually doing the transition at the rising edge of the clock. On the other end this second flip flop the flip flop 1, it makes it transition whenever Q 0 is equal to 1.

So, at this point Q 0 is equal to 1 now when the clock signal comes at this point Q 1 does not make a transition because Q 0 was equal to 0 at that time. Now so it could not make any transition, but at this point Q 0 was equal to 1. So, Q 1 makes a transition, so it becomes 1. So, after then again at this clock edge Q 0 is equal to 0. So, Q 1 does not transit, but at this point it will again transit.

So, if you look into this value that it is counting. So, it is 0 0 here then it is 0 1 at this point then it is 1 0 at this point, and then it is 1 1 at this point then it is again going back to 0 0. So, that is a that is a 2 bit synchronous binary counter. So, in this we can design synchronous counters also it is not mandatory that the counters should be asynchronous in nature.

So, you can design synchronous counters also the advantage with the synchronous counter is that the problem of glitch will not be there. So, asynchronous circuits we have those glitch, but in synchronous situations since we are bothered about changes only during the clock boundary. So, only when the clock edge comes the circuit will transit the flip flop will transit. So, the glitch problem will not be there, so that is the advantage and many of the counters are designed in a synchronous fashion.