# VLSI Physical Design

**Prof. Indranil Sengupta**
**Department of Computer Science and Engineering**
**Indian Institute Technology, Kharagpur**

## Lecture 24
## Clock Design (Part 1)

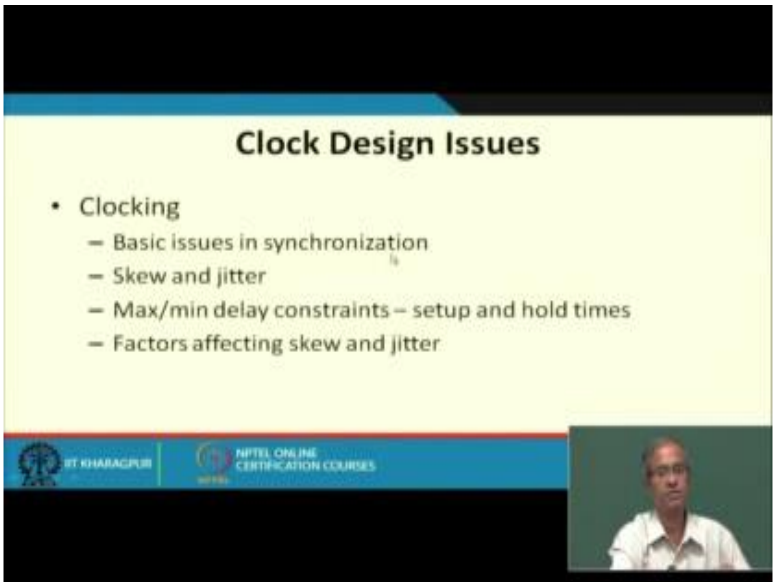
We shall now start on discussion on clocking in digital circuits. Now when over the next few week actually we shall be talking about various issues related to clocking and timing and cross talk and so on and so forth.

Clock is a very important component of any digital systems that we use today because, as you know most of the circuits and systems that we talk about that we use in practise are sequential in nature. So, there are lot of events which are going on inside the chip or the circuitry, and everything is controlled by a clock; a clock signal which acts as some kind of a synchronizing master for the entire system. Faster the clock faster would be the operation.

So, when we talk about clock there are a number of issues that we need to consider, of course, what is a clock signal? What are the different aspects related to the speed of operation of the circuit? What are the different kinds of parameters that you define with respect to a clock? And of course, we shall see later on that when we talk about high performance circuit which are very common nowadays we want to run a circuit as fast as we can; which means the delay of the circuit the clock frequency can be increased as much as possible. Of course, within some other limits of power consumption that will talk about later again.

So, we start our discussion in this lecture about Clock Design.

(Refer Slide Time: 02:17)



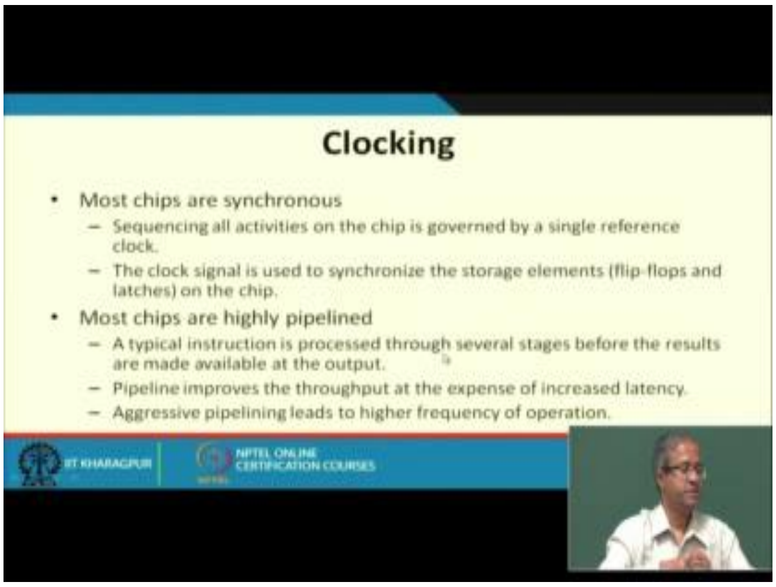
## Clock Design Issues

- Clocking
  - Basic issues in synchronization
  - Skew and jitter
  - Max/min delay constraints – setup and hold times
  - Factors affecting skew and jitter

As I said clocking is the basic concept behind synchronization in digital system. So, any sequential circuit all most all the sequential circuit today are synchronise in nature in contrast you can have a synchronise system which are of course much more difficult to design and control, although they might promise better or a higher speed it is extremely difficult to design and to verify such circuits and systems.

So, almost all the systems that we talk about in terms of synchronise circuits are synchronise in nature. There is a clock which controls all operations. So, there are a few very important clocking related parameters that we shall be talking about namely; skew and jitter. There are a number of delay constraints that also we shall talk about which are some kind of max constraints and min constraints these are called setup and hold times. And we shall talk about some of the factors that affect this skew and jitter kind of phenomena.

(Refer Slide Time: 03:37)

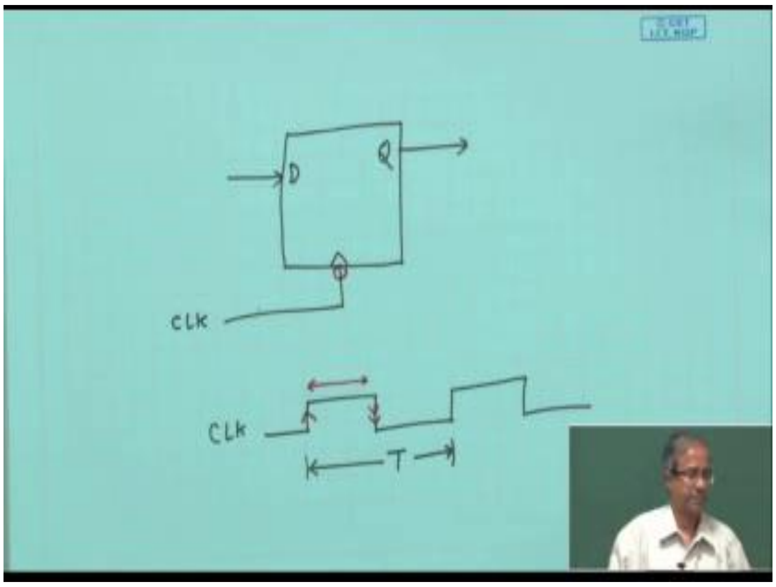


## Clocking

- Most chips are synchronous
  - Sequencing all activities on the chip is governed by a single reference clock.
  - The clock signal is used to synchronize the storage elements (flip-flops and latches) on the chip.
- Most chips are highly pipelined
  - A typical instruction is processed through several stages before the results are made available at the output.
  - Pipeline improves the throughput at the expense of increased latency.
  - Aggressive pipelining leads to higher frequency of operation.

So, this is what I talked about; most of the chips that are that are used today are synchronise in nature which means there is a reference clock typically there is a single clock in many systems there can be multiple clock phases which are generated from a single external source. So, all the internal activities of the chip they are synchronized with respect to the clock that is applied from outside typically. So, the clock signal is used to synchronize the storage elements, the flip flops because you know that when you have a flip flop for example.

(Refer Slide Time: 04:22)

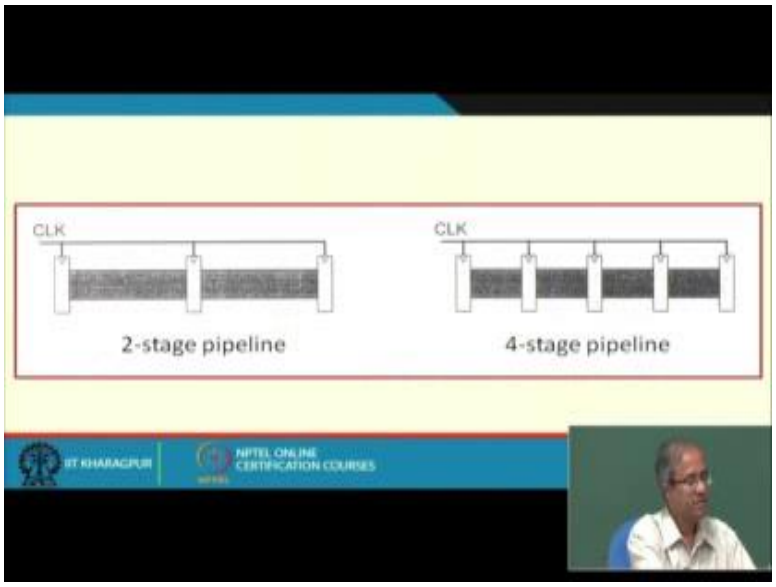


Let us say I have a D flip flop which has a D input and a Q output and there will be a clock input this is clock. So, here whatever I apply to D that will get stored inside the flip flop in synchronism with a clock. So, when I say the clock signal so the ideal clock signal will be like this it would be get periodic signal that we go up and down with certain time period let us say T. So, depending on the type of flip flop these flip flop can be activated either at the rising edge of the clock whenever the clock signal goes from 0 to 1 or whenever the clock signal goes from 1 to 0. So, when it is negative edge-trigged we usually use a bubble at the clock signal clock input to indicate this. These are typically called edge-trigged flip flop.

Now in contrast we can have a lache where this storage is not controlled by the edges of this clock but by the level, as long as the clock is high this flip flop or lache is open whatever is coming input in the at the input will get stored. And the last value, so after clock goes down that will be the value that will get stored finally.

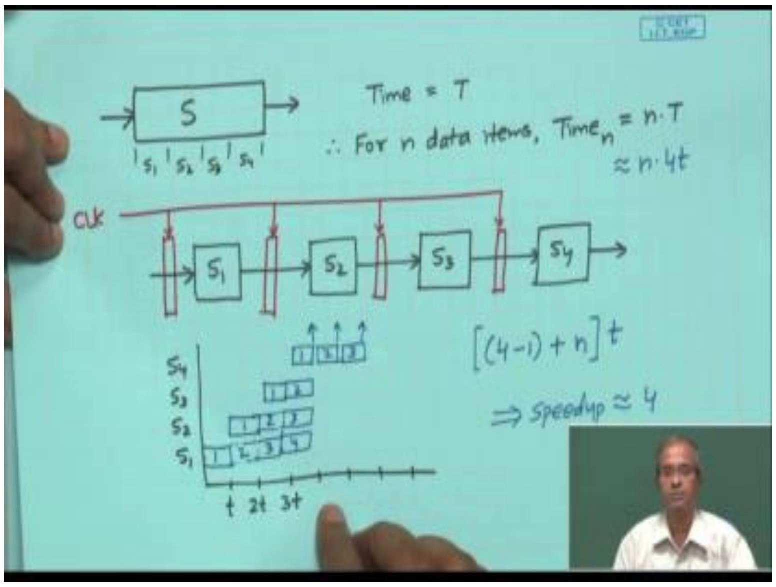
The next thing that we talked about is pipelining. Most of the system that we used today they are highly pipelined, they are pipeline systems or pipeline processors various kinds of. So, in a typical instruction pipelining for example; all the processors that we talk about today they use pipelining internally to improve the number of instructions that we executed per second it improves the throughput. And aggressive pipelining leads to higher frequency operations, I will try to explain this.

(Refer Slide Time: 06:44).



But how a pipeline looks like? A pipeline looks like something like this, this is an example of a two stage pipeline this black shaded regions are the two stages and there are some registers storage elements in between there are controlled by a clock, and this is a four stage pipeline. So, what is the basic idea behind the pipeline let me try to explain with the help of a simple this example.

(Refer Slide Time: 07:18)



Suppose I have some computation that I want to perform let us call it S. So, I apply an input I get an output. Suppose the time taken to process one set of data is T, therefore for n data items; so what will be the total time let us call it time n, this will be n multiplied by T. Now what I do, what I say is that instead of this single stage I divide this computation into smaller steps. So, let us take as example four. So, I call them S 1, S 2, S 3 and S 4. So, the input is coming the output of S 1 is going to the input of S 2 output of S 2 is going to input of S 3 and so on.

So, the idea is follows this whole computations S 1 I am as if dividing up into four parts; so I called them S 1, S 2, S 3 and S 4 it is not like I am multiplying hardware four times same hardware but I am partitioning it into four parts, which are approximately equal complexity in terms of time. Now in terms of data item let us say the input data are coming one two three four like that so it works like this, these are the times steps let say the time taken to for one stage to complete let us call it small t, so t 2t, 3t and so on.

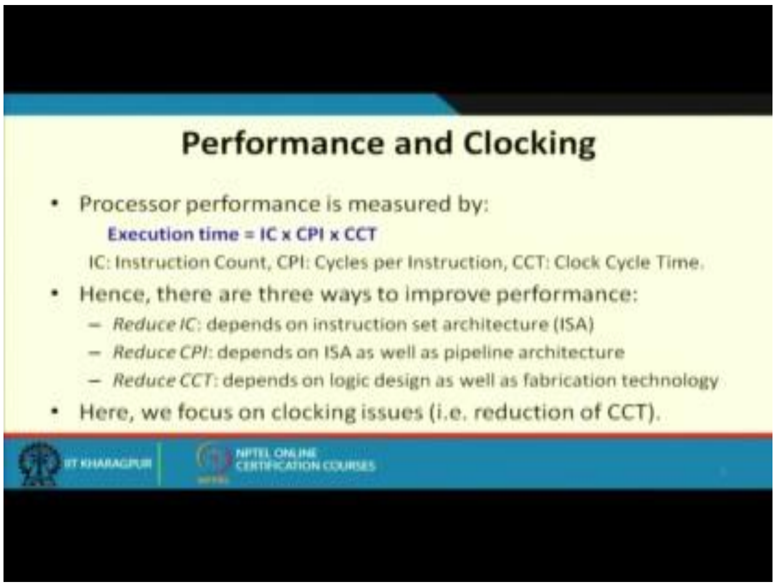
What will happen? The first let us indicate these stages S 1 S 2 S 3 and S 4. So, the idea will be like this; this S 1 will start working on data set 1, after it finishes it will go to S 2, n after it finishes it will go S 3, after it finishes it will go to S 4. Now when this S 1 output of S 1 goes to input of S 2 the idea of pipelining is I want to start the second input computation in S 1, in the same over lap fashion. Similarly when this goes to S 3 I want to start 2 here and start 3 here. So, in this way this will go on in a overlap fashion, in this way it will go on. You see after four times steps the first output is generated, and after that in every time steps one output will get generated.

So, for n data items you can have a calculation like this the total time will be number of stages here 4 minus 1, this is the time that is taken for the pipeline to fill up plus the number of data because after this initial time of 3 I will be getting 1; output every clock plus n whole multiplied by t. So, it is approximately n plus 3 right. See this n T you can say this is approximately equal to n into 4t. So, you can see in the same time I am able to have a speed up of about 4. So, this implies a speed up of approximately 4 equal to the number of stages, but of course in order to isolate these stages we need to have register stages in between the stages; this will be feed by a clock, this will a clock signal.

This is a basic idea behind pipelining, and almost all systems that we see today as an internal pipeline structure because of efficiency considerations, because of throughput increase in throughput that you get by doing that. So, this is the basic idea. You can see that may essentially we have several S 1, S 2, S 3, S 4 blocks which are combinational circuits and there are some registers or storage elements in between. This is how a typical circuit today looks like. So, this will motivate you into appreciate the examples that we that we shall see later on, because we shall see later on essentially what we have we have two storage elements with some combinational circuit in between.

So, in the lowest level there will be two flip flop with small combination circuit in between. So, if you can analyse that small circuit you can also analyse a pipeline kind of architecture which basically has a very similar structure, fine.

(Refer Slide Time: 13:11)



## Performance and Clocking

- Processor performance is measured by:
  **Execution time = IC x CPI x CCT**
  IC: Instruction Count, CPI: Cycles per Instruction, CCT: Clock Cycle Time.
- Hence, there are three ways to improve performance:
  - *Reduce IC*: depends on instruction set architecture (ISA)
  - *Reduce CPI*: depends on ISA as well as pipeline architecture
  - *Reduce CCT*: depends on logic design as well as fabrication technology
- Here, we focus on clocking issues (i.e. reduction of CCT).

So, another issue also let us also talk about. This is something related to the performance of a processor. So, here we are talking about a CPU, a processor which is executing instructions. Well, similar kind of discussion can apply to other kind of digital circuits also, but we are simply talking about processor here.

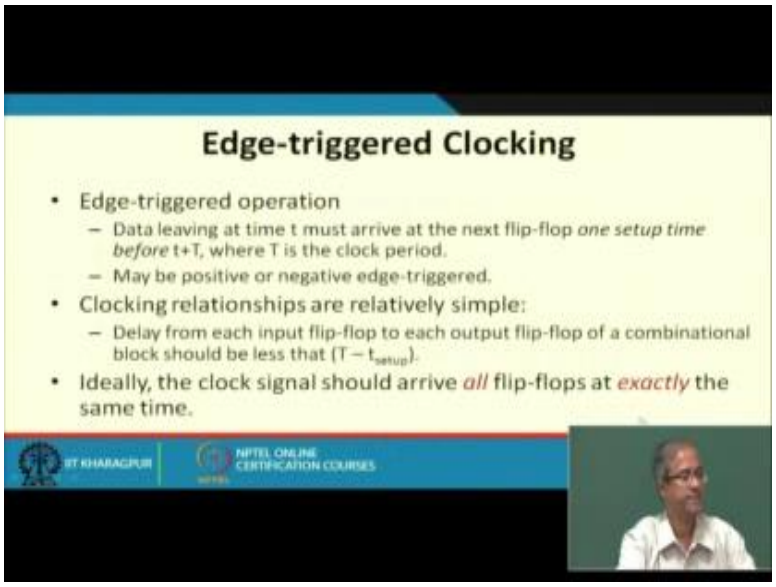
So, the total execution time here will be the product of number of instructions, instruction count, number of clock cycles required to execute every instruction this is called cycles per instructions multiplied by clock period clock cycle time. So, if you multiple this three things together you get the total execution time for a program. Now if I want to make a processor faster; that means I want to reduce the execution time; so how I can do it I can do it in three different ways: I can either reduce the number of instructions. This I can do by modifying the instruction set architecture.

Like for example, suppose I have a computer where I do not have a multiply instructions, so to multiply I may have to use a number of instructions. But if I move to a new instruction set where there is multiply number of instructions will be less; that is one way to reduce IC. Reduce cycles per instruction: well here we can have a better pipeline, deeper pipeline increase number of pipelines stages make the processing more overlapped will be getting greatest speed up or so. Number of cycle needed per instruction will go down.

And of course, when you talk about reducing the time period clock period or CCT it dependence on how you are doing logic design, this is something which will be the topic of our discussion primarily. And of course, on the fabrication technology; better technology will mean faster circuit which will mean faster clock. But technology is sometimes not within the control of a designer; designer can play with the logic design with the gates with the flip flops and find out ways to make circuit run faster without errors; that is what you want to look at.

So, our primary focus will be on the clocking issues specifically the third one reduction of CCT. So, how you can do this?

(Refer Slide Time: 15:48).

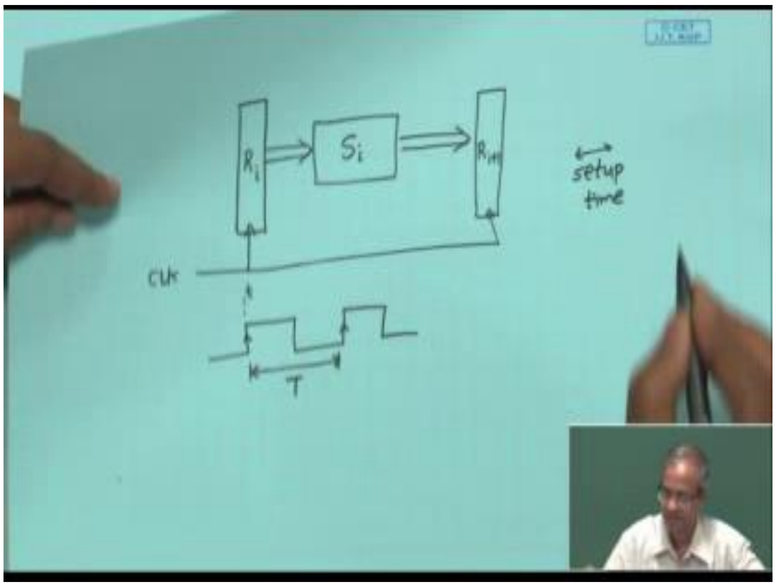


## Edge-triggered Clocking

- Edge-triggered operation
  - Data leaving at time t must arrive at the next flip-flop *one setup time before* t+T, where T is the clock period.
  - May be positive or negative edge-triggered.
- Clocking relationships are relatively simple:
  - Delay from each input flip-flop to each output flip-flop of a combinational block should be less that $(T - t_{setup})$.
- Ideally, the clock signal should arrive *all* flip-flops at *exactly* the same time.

Well, I talked about edge-trigged flip flop. There are a few statements you are making here of course will be explaining this little later in more detail. First is that most of the circuits that we use today there are based on edge-trigged operation. That means, the flip flop gets trigged whenever the clock goes from low to high positive edge-trigged or from high to low negative edge-trigged. So, data living at a time T let us say must arrive at the next flip flop. Well after t plus T; T is the clock period. There is something called once the consequence setup time this we shall see later, just let us ignore this for the time been now right now.

(Refer Slide Time: 16:50)



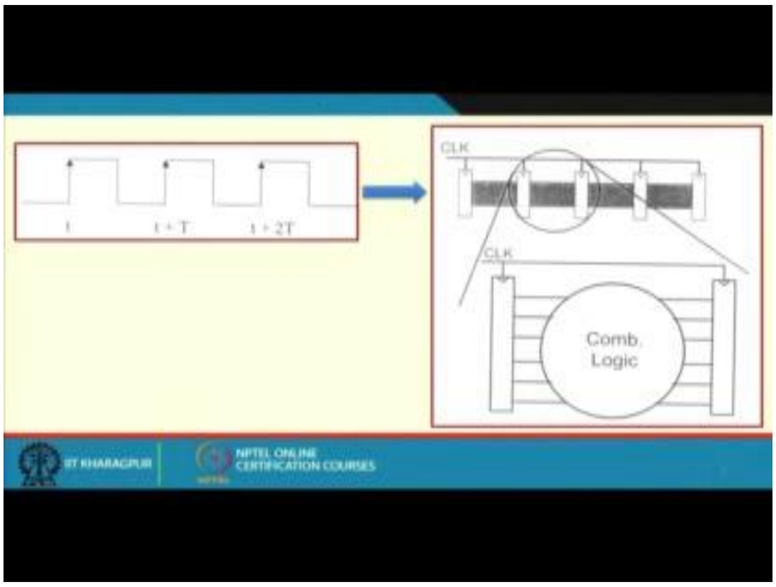
So what we are saying is that, data must arrive at the next flip flop one setup time before t plus T. Well here we are again talking about that pipeline kind of architecture. So, you stick to this kind of an architecture where I have stage which is a combination circuit, I have a register let us say R i, I have another register let us say R i plus 1. So, the output of this goes input of this output is goes to the input of this. And I have clock signal that is used to clock register as well as this register. And T is the time period of the clock. This is t.

So, what we are saying is that; whatever data that is living here let us say this time is small t so it should be ready, suppose these registers are all leading positive edge-trigged. So, when the next stage comes before that this data much be ready in the next flip flop so that the correct output will get stored. So, this capital time T; of course, there is something on setup time I shall talk about later, it is the properties of the storage elements. So, we have to give some you can say means provision with respect to the clock period to take care of this something called setup time this we shall see later; a little later.

So, basically the clocking relationships are like this; delay from each input flip flop to each output flip flop of combinational block, which means for a pipelines circuit the input register to the output register of a stage should be less than T, ignoring set up if you have setup. So, T will be become a little less. So, this we shall see little later why T is becoming little less. And another important property of the clock signals this also we shall be addressing later; that if there are many flip flops are register in circuit. So, what you want ideally is that the signal should reach all flip flops all most at the same time.

So, if this cannot be ensured then the correctness operations can be compromised. So, this is also one very important design parameter for clock circuitry that whenever I want to feed the clock signal they should all reach there almost at the same time. This is something we shall be discussing later, not right now.

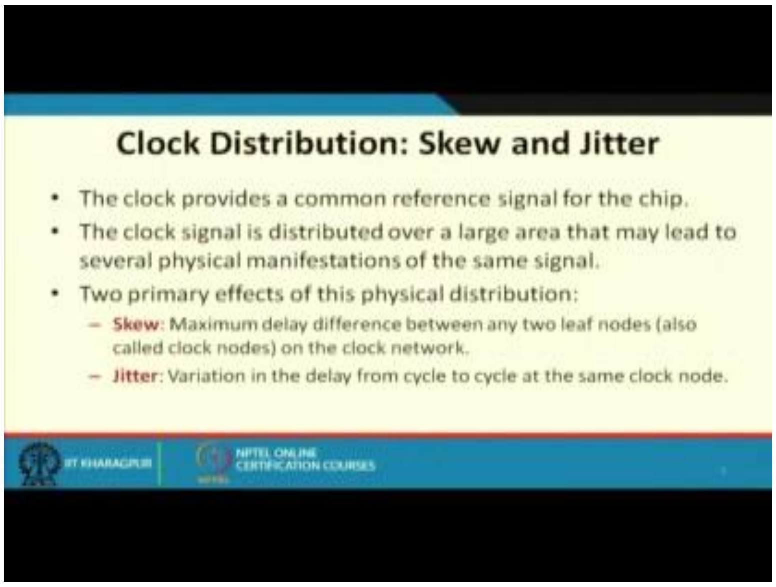
(Refer Slide Time: 19:43)



So, this is what we are saying that we have a clock signal coming starting from time T, t plus capital T is the time period t plus 2T. So, you have a stage let us consider one stage which essential is a combinational circuit, with a register before that, register after that, this clock comes. And this time T should be large enough so that whatever is the worst case of the delay of the combination circuit that computation should be complete so that whenever the next edge comes here the correct output should get stored here.

So, if this is ensured only then the pipelining should work properly that the output of stage 1 should go to stage 2, stage 2 go to stage 3 because, the next data is also coming parallely. So, there will be overlap kind of execution as data 1 goes from stage 1 to stage 2 data 2 will come into stage 2. So, there should be synchronization or a lock step movement of the data. If the clocking and delays are not absolutely synchronized there will be problem. If do not give sufficient time for a stage 2 finish it computation then may be the wrong output will get stored in the output register; that we should be very careful about.

(Refer Slide Time: 21:08)

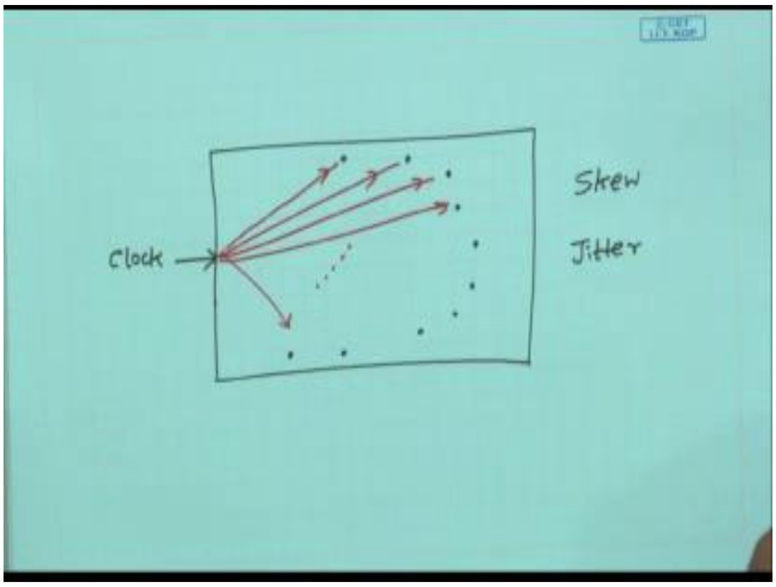


## Clock Distribution: Skew and Jitter

- The clock provides a common reference signal for the chip.
- The clock signal is distributed over a large area that may lead to several physical manifestations of the same signal.
- Two primary effects of this physical distribution:
  - **Skew**: Maximum delay difference between any two leaf nodes (also called clock nodes) on the clock network.
  - **Jitter**: Variation in the delay from cycle to cycle at the same clock node.

So, let us look at skew and jitter first. These are two parameters which relate to clock distribution. So, what do mean by clock distribution?

(Refer Slide Time: 21:24)



Clock distribution essentially means let us say I have a chip. As I said normally we apply a single source of clock from output, and inside there will be some circuitry which is allows this clock signal to go to several places in inside your chip where you need the clock signal to go. So, this clock signals will need to go here, it needs to go here, need to go here, to all this points. So, this is one property that this clock network should follow or ensure.

And another thing is that as I said that the clock signal should reach all the l points approximately at the same time, because if it is not ensured then there can be some error in computation. Let us see these two terminologies skew and jitter which are related to this distribution problem and how are they related.

So, as I said that the clock provides the common reference signal it may be distributed all throughout the chip, because it is distributed to all throughout the chip the total length of the clock net can be pretty large, and this will lead to several physical manifestations of the same signal. Various kinds of problems may arise because of this. So, what kind of problems? First is skew: skew says maximum delay different between any leaf nodes on the clock network. What does this mean?

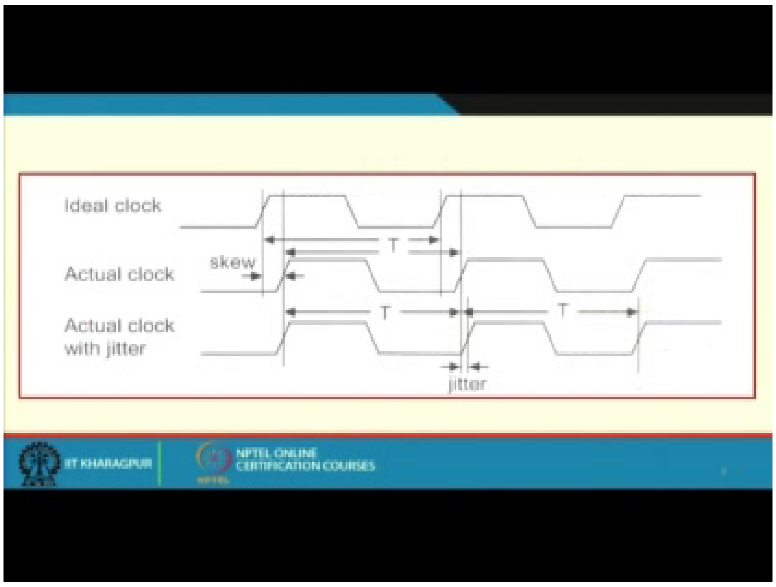
This is the clock network, the clock is coming here it must reach everywhere. Let us say this is the point x let us say this is the point y. So, what might happen is that in a point x the clock signal might be like this. So, what you may be see that because of difference in delay along different paths when clock reaches in y there can be delay like say the clock is delayed by some amount that depends on the addition delay on the path. And this is what is called as skew; the difference what the difference in the delays that whenever I am feeding a clock it is reaching the different end points of the leafs of the clock network at deferent times, this is what is called Skew.

Similarly, you can have jitter: let us say here I am looking at a particular point not both x and y, but particular point x. So, what I you can see is something like this let say I have a point x where the clock is coming, let us say I find that at the first clock is coming after a delay of T, the second clock is coming after delay of point 9 T, third clock signal coming after delay 1.1 T. So, the time period for the same leaf node the successive clock pulses are coming they should all be coming ideally with the period of T, T, T, T that gap, but there is a variation in delay across clocks across clock pulses that is called jitter.

Jitter is a variation in the clock cycle time for a single leaf node, and skew is across more than one leaf node there can be a variation in delay clocks can appear at various times.

These are the two problems which are important and if not handled or tackled properly the circuit might not work in a proper way. So, let us see.

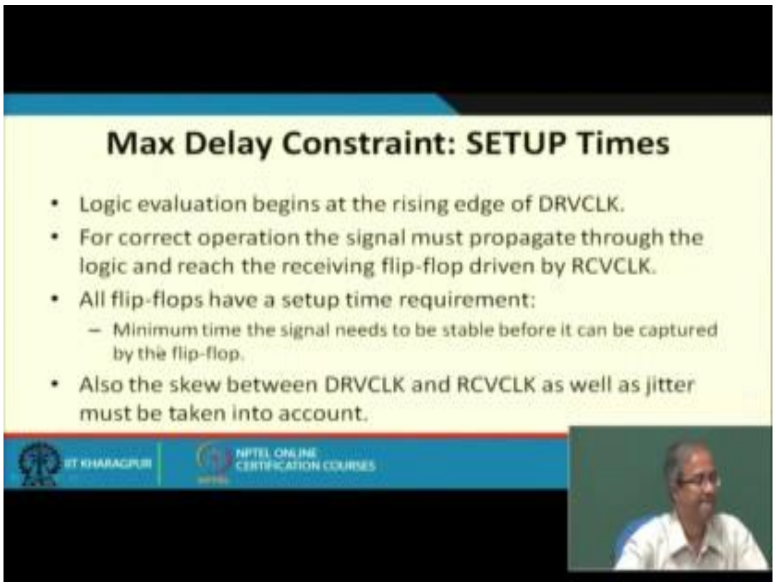
(Refer Slide Time: 25:45)



This is what skew and jitter as I have talked about they depict diagrammatically. So, you see this is a clock signal, where instead of ideal vertically rising vertically falling were showing slow rise and slow fall which are the typical shapes of the clock signals, because there will be a finite rise time and finite falls time because of resistive and capacity affects. And the actual clock what I am saying is that the time period is still T, but the edges are getting delayed.

So, the ideal clock should have come here but because of some additional delays the clock is reaching after some delay, this is called skew. But this skew delay is the same for all the successive pulses. But jitter says the first edge is coming after delay of this much the second is coming after delay of something more this plus this; this is called jitter because the clock edges are coming not with a delay of T where t plus jitter and the jitter can vary randomly; this is what jitter means.

(Refer Slide Time: 27:07)

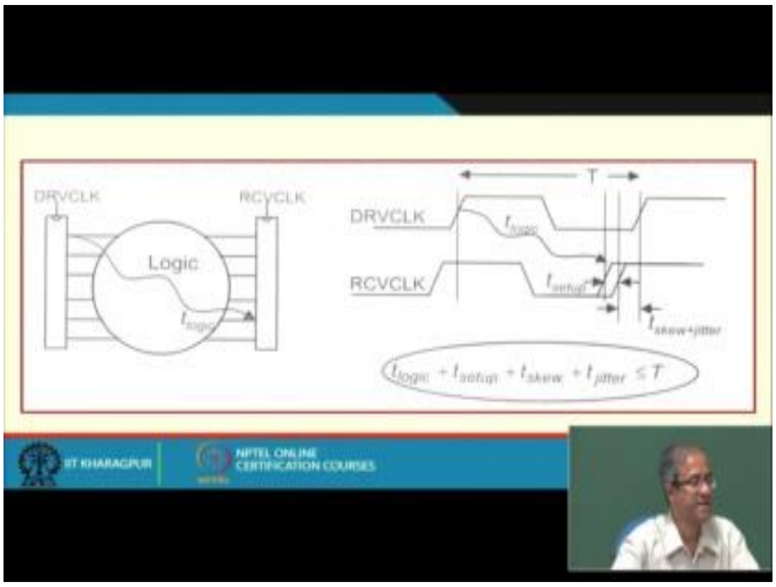


## Max Delay Constraint: SETUP Times

- Logic evaluation begins at the rising edge of DRVCLK.
- For correct operation the signal must propagate through the logic and reach the receiving flip-flop driven by RCVCLK.
- All flip-flops have a setup time requirement:
  - Minimum time the signal needs to be stable before it can be captured by the flip-flop.
- Also the skew between DRVCLK and RCVCLK as well as jitter must be taken into account.

Let us now look at something called setup and hold times that concerns storage elements.

(Refer Slide Time: 27:20)



So, what we look at is that a circuit like this let us say. Again I am talking taking example of a pipeline there are register stages and there is a combination circuits. So, let us name this two clock signal separately; I am calling this as driving clock, this is as receiving clock. And this is T logic is the delay through this logic circuitry or stage, let us see.

So, logic evaluation begin at the rising edge of driving clock why, because when we get the rising edge of driving clock that whatever is the input that is coming from the previous stage that will get stored in this register and this logic stage will start its calculations or computation. So, logic evaluation begins at that time.

So, for correct operation the logic signal or logic circuitry or must complete it evaluations before it reaches the flip flop and next stage of receive clocks comes. What I am saying is that, there is a delay of T logic and when this logic circuit has finish its calculation it must receive reach this point, and then only this clock should come otherwise some wrong value might get stored.

Now what we say, they say something called a set of time. So, every flip flop has some delay, delay with respect to what let a clock signal comes the input data does not immediately get stored in flip flop there is a delay after which the data gets stored. And also the input data has to be kept stable for minimum amount of time only then it will stored correctly, otherwise wrong data might get stored. This is the so called setup time.

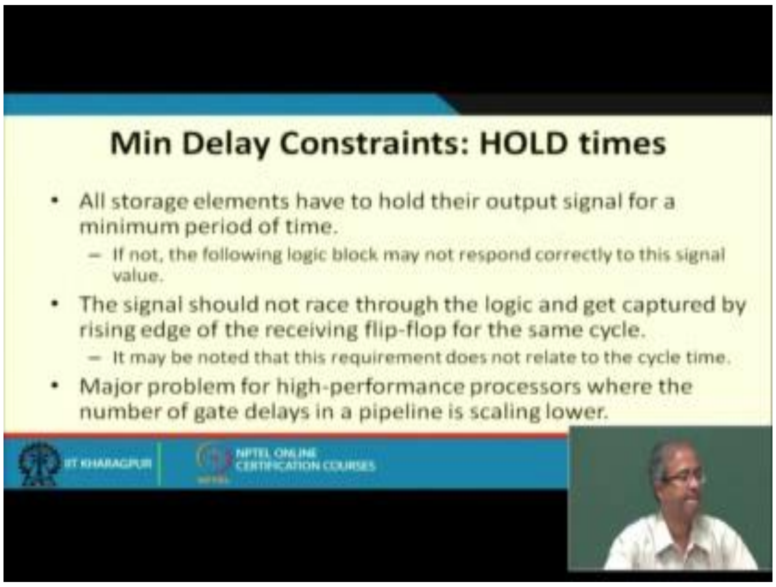
So, what the setup time defined as is? It is the minimum time the signal needs to be stable before it can be captured by flip flop. And when you are calculating the setup time you should also take care of this skew and jitter between the drive clock and receive clock. So, this we shall see slowly. So, look this timing diagram again; this is the drive clock, this is the receive clock. So, what we are saying is that; this is the delay of the logic maximum worst case delay of this stage.

So, from this edge whenever drive clock goes from 0 to 1 these stage the starts calculating. Suppose it takes this much time T logic is the total time when the data is available in the output of this logic circuit. After that I must keep this value stable minimum for this T setup amount of time, otherwise it not getting stored properly here. So, this is the requirement it says that you must wait for the maximum time the logic circuit requires to compute the data then you must wait for a minimum amount of time T setup which is the setup time of the flip flops here, then only we should apply receiving clock. The receiving clock edge should not come before this, if it come before this then there can be error.

And this time can further extended if we take into account skew and jitter, because if skew and jitter there can be some additional delays in the clock. So, receiving clock might get delayed. So, what will be your timing requirement? Your total time period should be greater than equal to the worst case delay of the logic circuit; that means, the stage of the pipeline plus this set time of the flip flop receiving flip flop plus the maximum clock skew plus the maximum jitter.

So, take you incorporate all of them add them up your time period should be at least greater than that, because whatever variations can be there and whatever minimum setup requirement is there you must provide that much time otherwise the correct data may not get stored in the receiving flip flop, right?

(Refer Slide Time: 32:02)



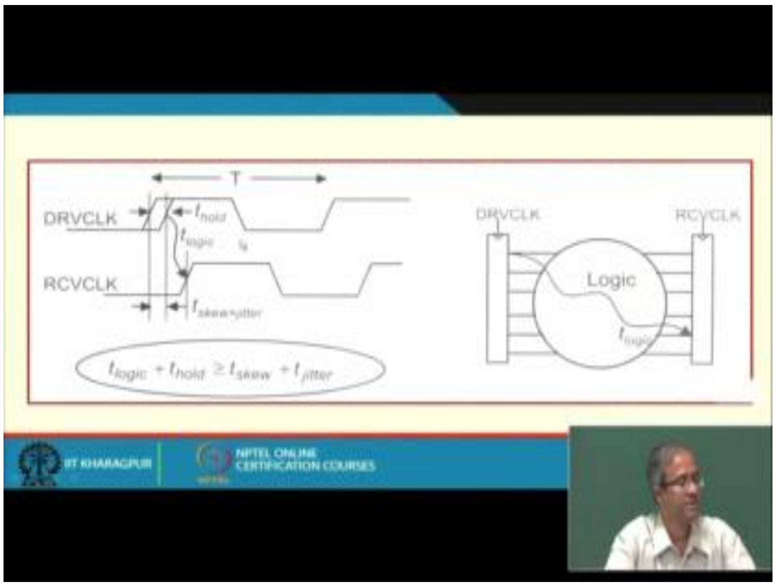
## Min Delay Constraints: HOLD times

- All storage elements have to hold their output signal for a minimum period of time.
  - If not, the following logic block may not respond correctly to this signal value.
- The signal should not race through the logic and get captured by rising edge of the receiving flip-flop for the same cycle.
  - It may be noted that this requirement does not relate to the cycle time.
- Major problem for high-performance processors where the number of gate delays in a pipeline is scaling lower.

So, in a similar way you can have another constraint that is called a min delay constraint. You see setup time was what? Setup time says that you have a flip flop data is coming in the input, your input data must be stored for a minimum amount of time before the clock comes that minimum amount of time is setup time.

Now, hold times says the other around; after the clock edge comes you have to wait for some more time. See it says that this storage element will have to hold their output signal for a minimum period of time.

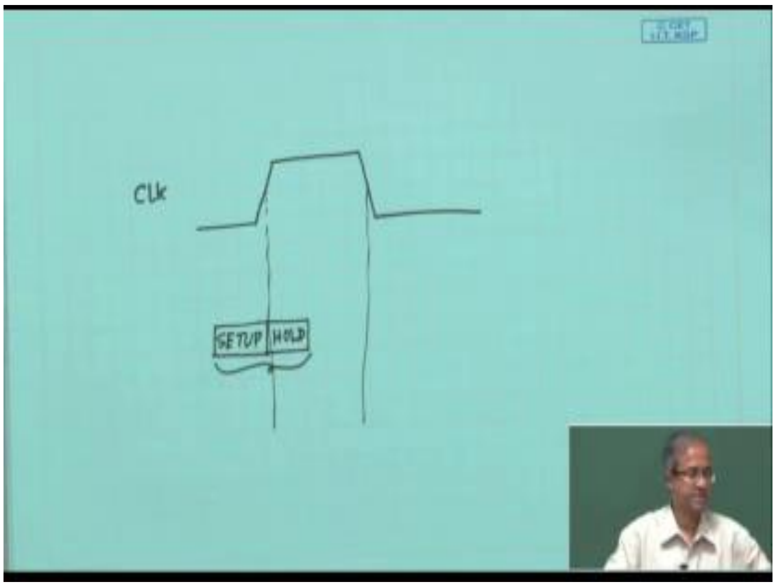
(Refer Slide Time: 32:49)



So, if you do not do it then the following logic block like a same kind of scenario driving clock receiving clock we shall be explaining with this. So, this hold times says this is the definition. So, the ideal is that signal should not go through the logic circuit too fast and get captured by the rising edge of the receiving flip flop of the same cycle. This is something let me let me explain this example again; take this example.

So, I have this driving clock, driving clock it comes T is the time period. So, what are the timing requirements? After the driving clocks come the hold time you must, this hold time is the time you must keep the input data stable before the clock comes. And let me talk like this.

(Refer Slide Time: 34:00)

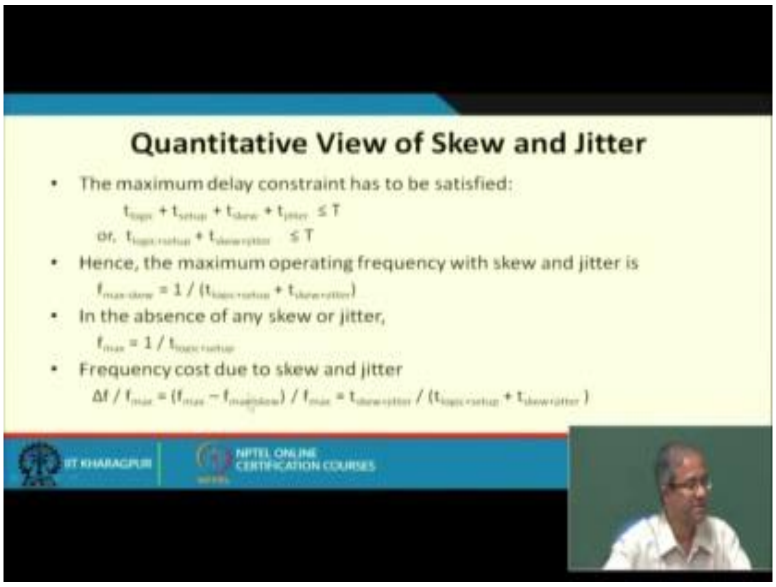


Because I have clock like this it is explain here this is the clock edge, this is the clock edge. So, what I am saying is whenever I have a data coming, so this is time the clock edge is coming; what I am saying? I must keep my data stable minimum time before this is my setup time, and after the edge comes I have to again wait for a minimum time that is my hold time. These two times I must provide then only my flip flop operation will be guaranteed to be faithfully correct.

So, some minimum time before the edge, some minimum time after the edge these are setup and hold. You see in this diagram we have illustrated the hold time. After the edge comes a minimum amount of T hold time must be provided, and only after that the logic calculations (Refer Time: 35:06). So, what I am saying is that whenever the driver clock comes so earlier we have assumed that whenever the driver clocks comes the logic operation start calculating immediately. But now we are saying no, not immediately after the clock comes let us wait for another time T hold before this logic operations is allowed to start because, that time T hold is required for the output to stabilise.

So, whenever the clock comes T hold is that time that is required then only this T logic will coming this is the T logic delay, and then you take care of this skew and jitter. So, T logic plus T hold is the total time; T logic is the delay of this logic and T hold is a minimum time we should wait before we should start the calculation. This two taken together should be greater than this time. If this is not ensured this operation of this circuit might fail.

(Refer Slide Time: 36:23)



## Quantitative View of Skew and Jitter

- The maximum delay constraint has to be satisfied:

  $t_{logic} + t_{setup} + t_{skew} + t_{jitter} \leq T$

  or, $t_{logic+setup} + t_{skew+jitter} \leq T$
- Hence, the maximum operating frequency with skew and jitter is

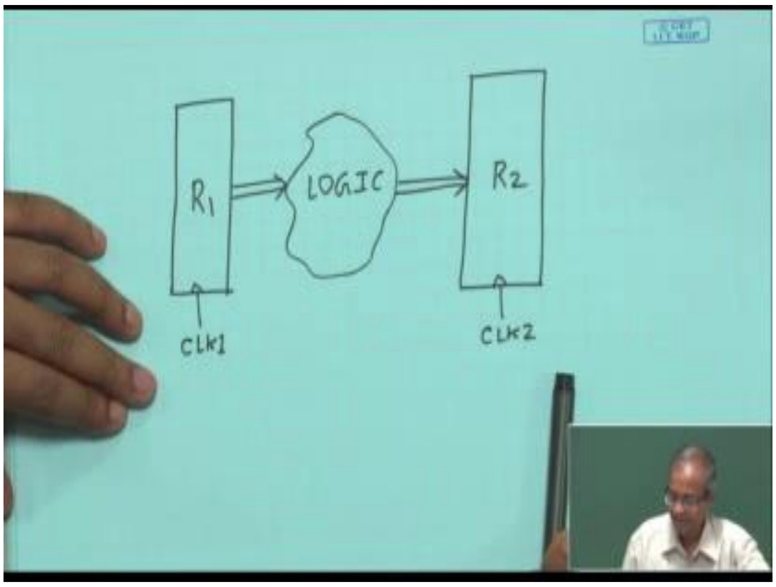
  $f_{max-skew} = 1 / (t_{logic+setup} + t_{skew+jitter})$
- In the absence of any skew or jitter,

  $f_{max} = 1 / t_{logic+setup}$
- Frequency cost due to skew and jitter

  $\Delta f / f_{max} = (f_{max} - f_{max-skew}) / f_{max} = t_{skew+jitter} / (t_{logic+setup} + t_{skew+jitter})$

Now, let us look at some of the quantitative views of skews and jitter whatever we have talked about so far. There are a few constraints that need to be satisfied; let us look at one by one.

(Refer Slide Time: 36:43)



Again we consider the same kind of the scenario: there are two register stages and there is a logic circuit in between, this is logic, this is a storage element R 1, this is another storage element R 2. So, they are connected like this. So clocks, well this clocks are same clock but I am showing them separately because there can be some skew and jitter between them, so there can be some delay between clock 1 and clock two because of skew and jitter. So, we have a scenario like this, so we have now looking at skew jitter setup and hold times there are a few constraints that need to be satisfied for correct operation of the circuit. Let us summarize the constraints once more.

So, the first constraint is like this; this is the maximum delay constraint what it say is we told you about, the total logic circuit delay the setup time delay then you take care of this skew and jitter of the receiving clock, your time period should be at least greater than equal to this. Now logic and setup if you combined together let call a T logic plus setup, skew jitter if you combine together call it skew plus jitter, they should be less than equal to T. This is one constraint. Setup time is what? Again I am repeating before clock how much time you have to wait? You have to feed the data and keep it stable, that much time.

So, here when you are talking about a scenario like this, this logic circuit is feeding data to in next stage so that setup time will apply to R 2 now. When data is coming here the data must be kept stable for minimum amount of time T setup before the clock here comes. Similarly for R 1, this stage before that whenever the data come there will be component T setup here; so logic plus T setup plus that jitter and skew that whole thing that should be taken care in that time period T; this is the first constraint.

So, with this you can say that the maximum operating frequency with skew and jitter will be 1 by this, this will be the minimum delay. So, if you take the reciprocal of this 1 by this, this will be the maximum frequency with this circuit can operate. And if you ignore this skew and jitter for the time being theoretically the maximum frequency would have been just the logic and setup times this 1 by T logic setup time.

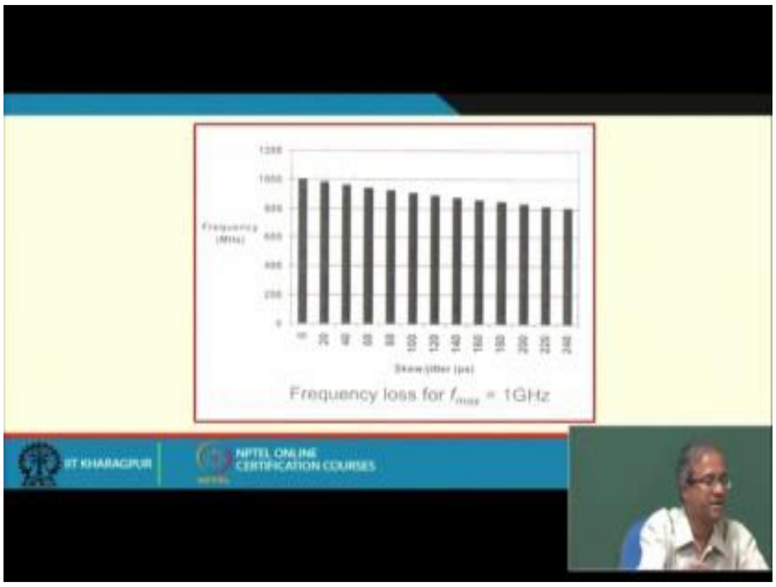
So now, how much frequency you are losing? You see now we are looking from a design point of view, see as a designer I have designed a circuit, I want my circuit to run as fast as possible so how do I measure the fastness of my circuit, the maximum clock frequency. So theoretically, well initially I did not thought about skew and jitter. So, just assuming that clock are coming and getting distributed very nice way I have estimated that where my maximum frequency should have been this, but after designing my circuit because of skew and jitter what I find that I cannot go up to that maximum frequency, because if I do that due to skew and jitters some of the stages might not stored the data correctly. So, I have to delay the clock a little more.

So, the actual frequency has to be reduced a little bit; that is the price I am paying. So, here in absence of skew and jitter I have this. You can say that your incurring something call a frequency cost; what is frequency cost? It is the difference in the frequency the maximum frequency that you could have achieved if skew and jitter where not present minus the actual frequency in place of skew and jitter divided by f max. So, if you do a simple means arithmetic on this the expression come to T skew jitter plus T logic setup plus T skew jitter.

If you can reduce skew jitter as much as possible, this frequency cost will also reduced accordingly. One of the main challenge for the designer is; is how to design the clock circuit, how to design the clock network such that your skew and jitter are minimised: because you can say that well skew and jitter how this can be under my control because once I fabricated a chip I do not know how much delay it will be taking, but at least you can try, you can try to ensure that the length of all the clock wire are approximately same so that the estimated delay should be the same.

So, the delay variation should be minimum, this skew should be less. This is what you can expect. So this is what, and just one sample data I am showing.

(Refer Slide Time: 42:34)



Frequency loss for $f_{max}$ = 1GHz

So, for maximum frequency of 1 gigahertz this is one sample plot which I am showing; that as this skew jitter increases from 0 this numbers are picoseconds up to 240 picoseconds you can see the frequency drops from 1 gigahertz down to about 800 megahertz. So, as skew jitter increases your maximum frequency also goes down. One thing that will be discussing in our subsequence classes, in our clock design, clock tree design in this kind of lectures there we shall see that how you can design our clock circuitry in such a way that this jitter and skew can be reduced as much as possible, because that is the very important component to reduce the frequency cost to the extent possible.

So, whatever I have talked about today in this lecture we have tried to introduce the concept of frequency cost means, how much frequency we have to sacrifice; that means, the maximum speed of operations in presents of skew and jitter. The other parameters are something which we do not have any control about. For example, hold time and setup time these are characteristic of the flip flop, the storage elements, those time we have to provide, those are mandatory delay requirements, but skew and jitter is something which you can play around with.

So, we shall be looking at some more examples in our next lecture so that you can have better feeling about how we can do some kind of trade out in terms of delays in other parameters so that you can avoid some time violations, you can you can able to run your circuit in higher speeds and so on. With this we come to the end of this lecture.

Thank you.