Microelectronics: Devices to Circuits
Professor, Sudeb Dasgupta
Department of Electronics and Communication Engineering
Indian Institute of Technology Roorkee
Lecture 58
Clocking Strategies for Sequential Design

Hello everybody and welcome once again to the next addition of NPTEL online certification course on Microelectronics Devices to Circuits. In a previous module we have looked into the concept of combinational sequential logic and we have also understood that is equation logic is basically logic where memory is inbuilt into it and we also saw that unlike combinational, in sequential logic timing is a very important issue.

So we should be able to therefore perceive that at a particular rising edge of the clock if my data has to be inserted into the system then how will the sequential logical block react if the data would have inserted much earlier than the rising edge or much later than the rising edge and what would be the effect on the overall performance of the sequential logic. Keeping in mind this module is basically, if you look is termed as clocking strategies for sequential design.

(Refer Slide Time: 1:33)

IIT ROORKEE

NPTEL ONLINE
CERTIFICATION COURSE

MICROELECTRONICS: DEVICES AND CIRCUITS

CLOCKING STRATEGIES FOR SEQUENTIAL DESIGN

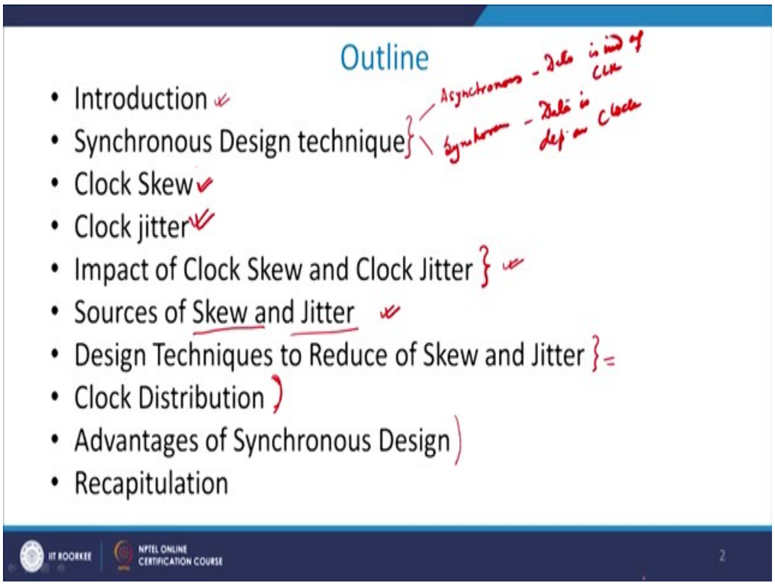
SUDEB DASGUPTA

DEPARMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

1

So this is a topic of this module and outline of this module is as follows.

(Refer Slide Time: 1:40)



Outline

- Introduction
- Synchronous Design technique
- Clock Skew
- Clock jitter
- Impact of Clock Skew and Clock Jitter
- Sources of Skew and Jitter
- Design Techniques to Reduce of Skew and Jitter
- Clock Distribution
- Advantages of Synchronous Design
- Recapitulation

Asynchronous - Data is ind of CLK

Synchronous - Data is dep on Clock

IIT ROORKEE
NPTEL ONLINE CERTIFICATION COURSE
2

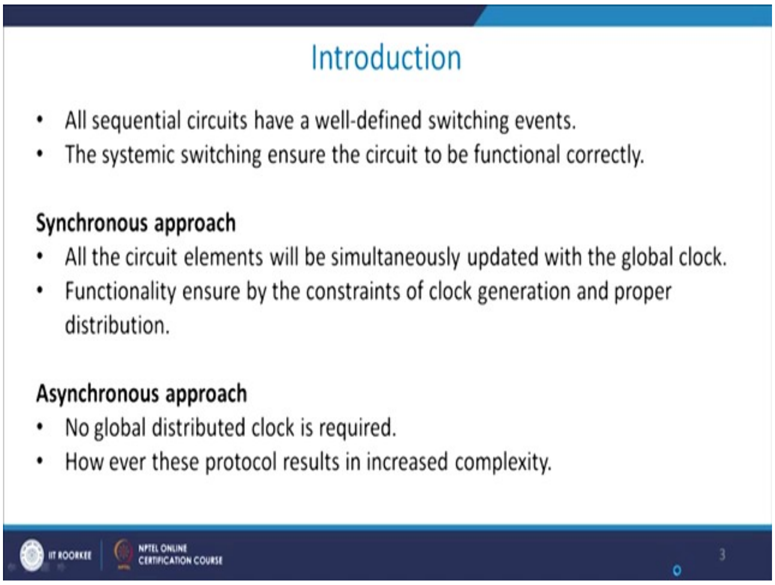
We will first introduce to you the basic concept of timing and then we will look into the synchronous design techniques. As they have discussed earlier there are 2 types of design, one is A synchronous, right? And we have another is synchronous. What is A synchronous? Well, Asynchronous is that when the data is independent of clock, right? External clock and when data is dependent on clock we define that to be as Asynchronous design.

So for Asynchronous design timing issues are not important but for synchronous design timing issues with respect to clocks are very important. In these respects we look into 2 major problem areas of clock and one is the clocks skew, another known is clock jitter and we will see how this influence the overall frequency of operation of a sequential logical block. And that is what is the fifth point here is all about that its impacts on the overall delay of the system.

We will also look into the fact from where skew and jitter do come into picture because if you know the sources from where it is coming we should be able to therefore reduce the skew and jitter here, right? So we would look into what is the meaning of skew and jitter. The origin of skew and jitter and then ways and means of reducing the values of skew and jitter. Then we will have a look at the clock distribution problem and then advantage of synchronous design as compared to Asynchronous design then finally we will recapitulate the whole thing.

So as far as this module is concerned, it gives you typically a brief idea about the locking strategy.

(Refer Slide Time: 3:26)



Introduction

- All sequential circuits have a well-defined switching events.
- The systemic switching ensure the circuit to be functional correctly.

Synchronous approach

- All the circuit elements will be simultaneously updated with the global clock.
- Functionality ensure by the constraints of clock generation and proper distribution.

Asynchronous approach

- No global distributed clock is required.
- How ever these protocol results in increased complexity.

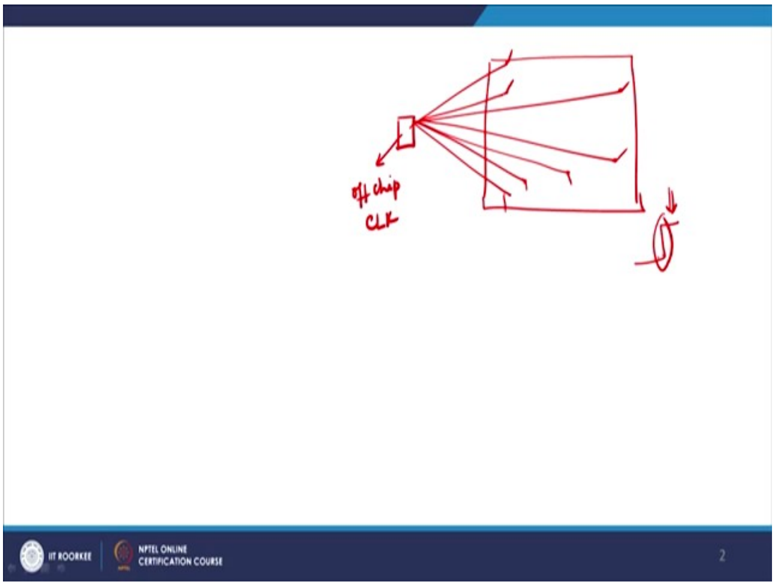
Now let me come to the basic approach and we will go step-by-step explaining to you each one of them. So as I discussed with you all your sequential circuits, right? Have well-defined switching events. As I have discussed with you which means that these switching systems ensure that the functionality of the circuit is not compromised. As I discussed with you earlier also that let us suppose it is a positive edge triggered register maybe which is accepting a data and then it is giving to a combinational logical block where some computation is being carried out maybe a logical operation and after that it is fed into another register where it is storing for a few amount and then giving it to the output side.

Now the way the data will travel from input to output through these 2 registers as well as combinational logical block, we will determine the speed of the clock. Maximum speed of the clock. Now if for example I have discussed with you that if the input is much much earlier than the setup time of the register then that data will not be at all inserted into the system and you will get a miss.

Similarly if the data does not hold its value beyond a particular limit after the clock is passed which is known as t_{hold} you will not be also able to evaluate the system. So that is the reason why we define that sequential logical are well-defined switching events. Based on switching events. Now there are 2 approaches as I discussed with you in the starting slide we have got synchronous approach and we have got Asynchronous approach.

In this case as I told you all the circuit elements will be simultaneously updated with the global clock which means that there will be a global clock which will be driving all the blocks digital block within the system and this global clock assuming that the global clock is being run in such a manner that the time at which it goes from 0 to 1, rising edge of the clock is exactly the same in all the blocks then we will accept that and all the blocks the data will be inserted at the rising edge.

(Refer Slide Time: 5:41)

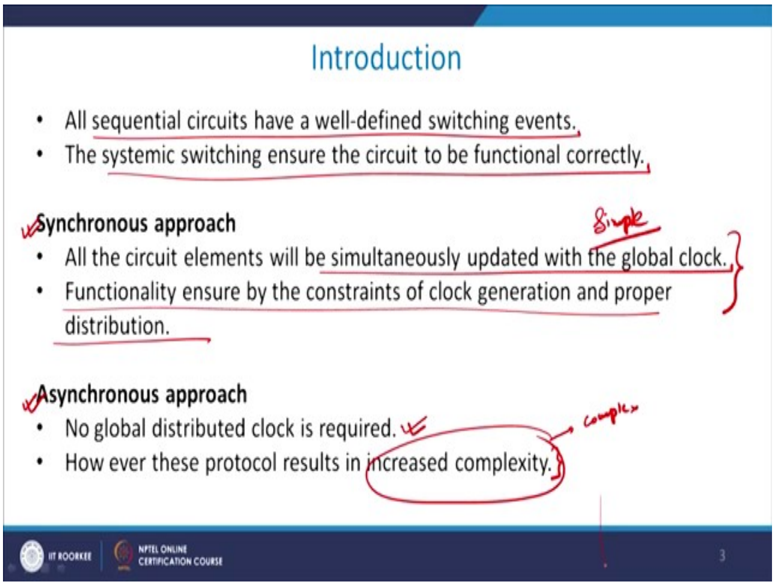


I will give you an idea let us suppose what we are assuming, we will see that this assumption is wrong later on but for understanding purposes, let us suppose you are driving and off chip clock. So this is my of chip clock here, right? And it is driving through this interconnects to the various parts of the sequential block here, right? So these 2 are almost equal in diamonds.

This is the farthest, these 2 are farthest and this is somewhere in the middle which you see. So somewhere here also you can get. Now I am assuming that when there is a rising edge of the clock this event is replicated at this point, does point, this, this, this, this and this, which means that irrespective of the routing interconnect between the clock and the area where you are giving the clock irrespective of the distance mode.

I assume that the clocks always at 0 to 1 transition at t equals to 0 let us suppose, right? And that is a major assumption we will break off this later on.

(Refer Slide Time: 6:44)



Introduction

- All sequential circuits have a well-defined switching events.
- The systemic switching ensure the circuit to be functional correctly.

Synchronous approach

- All the circuit elements will be simultaneously updated with the global clock.
- Functionality ensure by the constraints of clock generation and proper distribution.

Simple

Asynchronous approach

- No global distributed clock is required.
- How ever these protocol results in increased complexity.

complex

IIT ROORKEE | NPTEL ONLINE CERTIFICATION COURSE

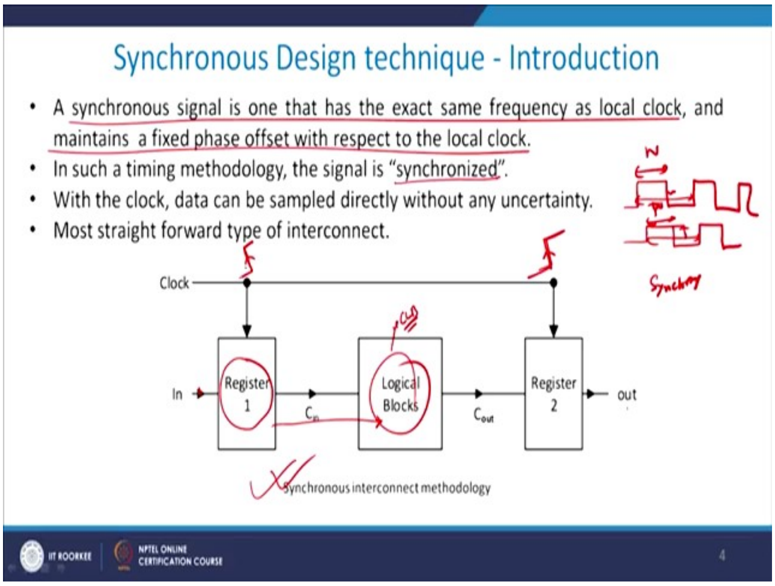
3

So that is one important clock that there will be a global clock. And therefore the functionality will ensure that by the constraints of clock generation and proper distribution. As I discussed with you therefore that clock going into the system and generating an output logic will depend therefore on the proper synchronization of the clock with respect to individual blocks of the sequential data. This is for the synchronous approach.

You can also have an Asynchronous approach which is there. Asynchronous approach means that there is no global distribution of the clock. So there is no clock is required, however, so it depends upon what? It depends only upon the rate of flow of data directly coming into the system, however in these cases there is an increase complexity which is with us.

The complexity is typically very large, they are complex circuits, whereas these are simple circuits where you need just to rout the signal at a particular point. So this is the basic introduction of a clock.

(Refer Slide Time: 7:47)



Synchronous Design technique - Introduction

- A synchronous signal is one that has the exact same frequency as local clock, and maintains a fixed phase offset with respect to the local clock.
- In such a timing methodology, the signal is "synchronized".
- With the clock, data can be sampled directly without any uncertainty.
- Most straight forward type of interconnect.

Synchronous interconnect methodology

Now let me therefore define certain definitions I will explain to you. Asynchronous signal is one that has exactly the same frequency as the local clock and maintains a fixed phase offset with respect to the local clock. Which means that if I have a local clock which is coming like this, this is a local clock than if I were clock which is something like this, so this is T, this is also T.

Say this is width this is width, this is T of the main clock this is also T of the main clock but it suffers from a basic phase difference here then we define this to be as Asynchronous signal. Signal is synchronous. The signal is said to be synchronized. As I discussed with you, why synchronized? For example if you look back into this design here in front of you. There I have a register here, clock here.

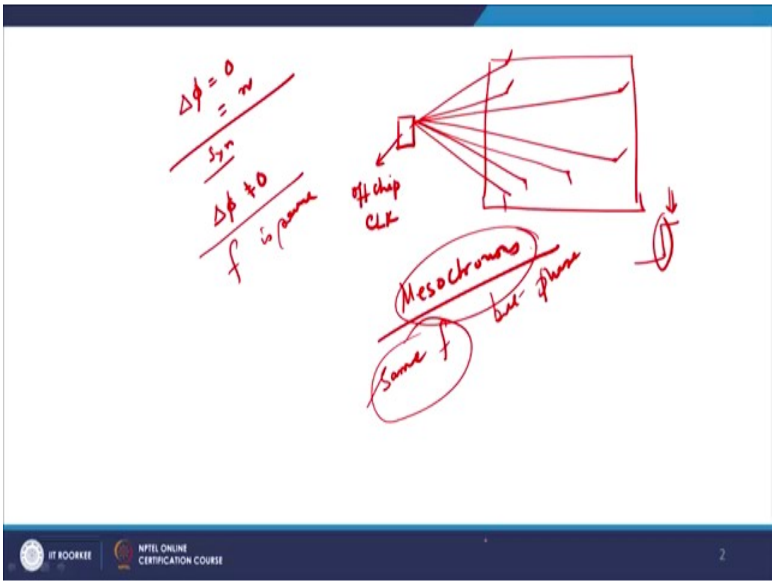
I am assuming that the clock transition, these are all edge triggered register suppose. So whenever my edge goes high, it also goes high. And this register is able to insert one data from the data train here and transfer it to the logical block. So this is your combinational logical block which does all sorts of computation, a logical block and gives you C_{out} goes to the register 2 and that is output.

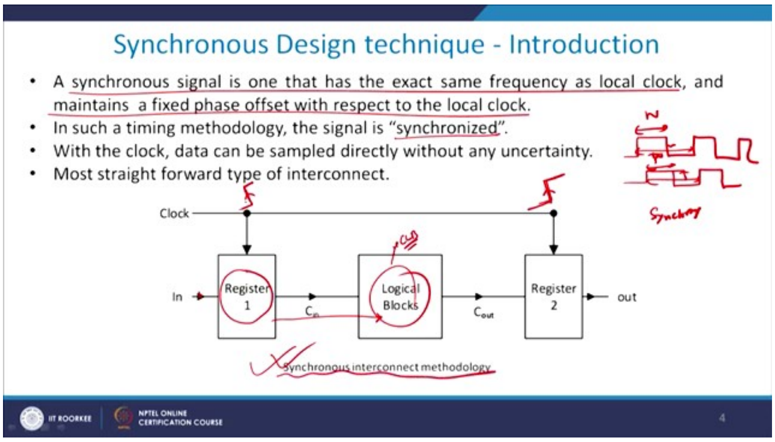
So which means that, if I assume that the logical block has got 0 delay then I would expect that the insertion of data at register 1 and output at register 2 are exactly the same time domain provided a delay is 0 for the combinational logical block. In reality not true and therefore that gives you a constraint on the total time period of the external clock for sequential logic.

Now with the clock as I discussed with you data can be sampled directly without any uncertainty and so this is the most straightforward simplest interconnect design for using your synchronous digital technique and it is pretty easy to implement also.

There is also another 2 types of important designs which is there with us. One is known as Mesochronous us and other is known as plesiochronous.

(Refer Slide Time: 10:10)





Synchronous Design technique - Introduction

- A synchronous signal is one that has the exact same frequency as local clock, and maintains a fixed phase offset with respect to the local clock.
- In such a timing methodology, the signal is "synchronized".
- With the clock, data can be sampled directly without any uncertainty.
- Most straight forward type of interconnect.

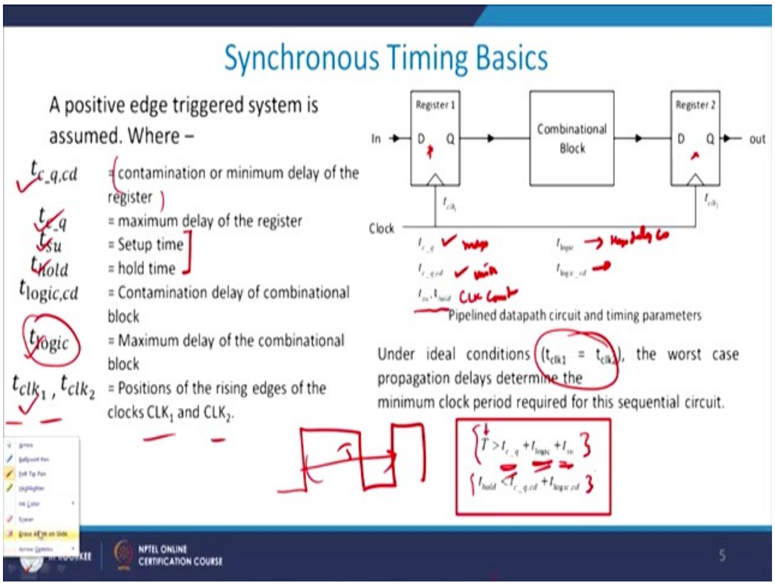
Synchronous interconnect methodology

So I have got 2 type of designs one is known as Mesochronous. In a Mesochronous, synchronous we have already discussed. Exactly in phase or equal phase and they are same frequency. Mesochronous have the same frequency, so 2 clocks, so I have a master clock and I have a clock, I have a data train then the data train is said to be Mesochronous provided they have the same frequency but we cannot say it about the phase. So phase will be varying.

In the first synchronous case, you will have the same frequency of course but the phase will be fixed. Phase difference will be fixed, so the phase difference Δϕ will be either 0 or some value, integral value N and it will remain fixed for all the edges or all the designs and therefore this is known as synchronous, right? Whereas let us suppose Δϕ is not equal to 0 and it remains like this for a period of time we will define that to be as a Mesochronous though your frequency is same, fine.

Plesiochronous basically means that neither this frequency nor the phase is same in both the cases, so there are 3 types of interconnects models available we will not go into details of that one at this stage but I will give you a brief idea later on if time permits. So this is basically synchronous interconnect methodology which is in front of you and gives you quite a good definition of synchronous timing.

(Refer Slide Time: 11:38)



Synchronous Timing Basics

A positive edge triggered system is assumed. Where –

$t_{c\_q,cd}$ = contamination or minimum delay of the register

t_{c_q} = maximum delay of the register

t_{su} = Setup time

t_{hold} = hold time

$t_{logic,cd}$ = Contamination delay of combinational block

t_{logic} = Maximum delay of the combinational block

t_{clk_1}, t_{clk_2} = Positions of the rising edges of the clocks CLK$_1$ and CLK$_2$.

Pipelined datapath circuit and timing parameters

Under ideal conditions (t_{clk1} = t_{clk2}), the worst case propagation delays determine the minimum clock period required for this sequential circuit.

$$\boxed{\begin{array}{l} \{T > t_{c\_q} + t_{logic} + t_{su}\} \\ \{t_{hold} < t_{c\_q,cd} + t_{logic,cd}\} \end{array}}$$

Now let me come to the basic timing analysis, we have already done this in our previous class but I will still like to give you an inside window once again, so the first delay which you see is the setup and whole time, very simple. Setup time is defined as the time before the rising edge of the clock when the data should be held stable, so that the input register is able to properly accept the data.

What is hold time? Hold time is the minimum time after the rising edge of the clock, same clock till which the data must be held stable for the clock to understand the fix data has come to our system. Now maximum delay of the combinational logical block is known as t_{plogic} or even t_{logic}, t_{clk1} and t_{clk2} are the positions of the rising edges with respect to clock 1 and clock 2, so we have done with this, we have done with this understood setup and hold time.

Now the maximum delay of these registers this or this is referred to as $t_{c\text{-}q}$. Now $t_{c\text{-}q}$ is defined as the contamination or minimum delay of the register. So $t_{c\text{-}q}$ is the maximum delay of the register whereas $t_{c\text{-}q,cd}$ is basically defined as the minimum delay of the register and $t_{logic,cd}$ is the combinational logical block error in terms of the delay, higher delay higher error will be there.

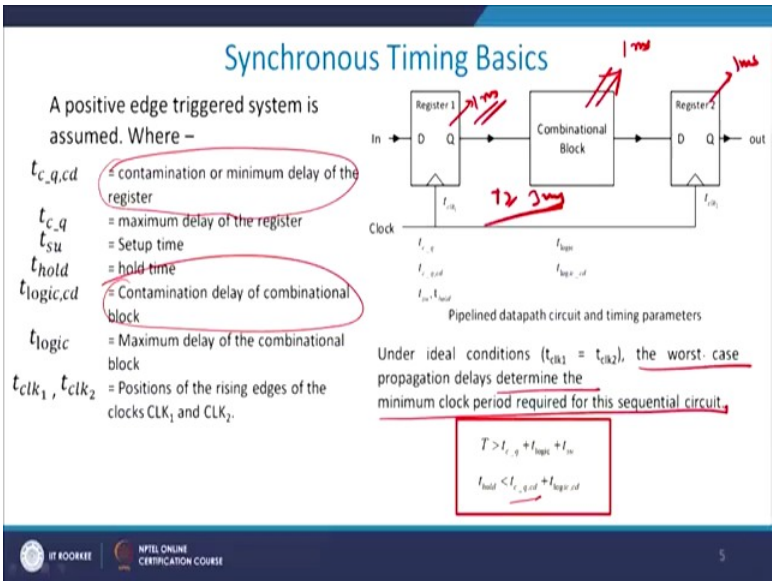
So we have got 5 quantities here t_{cq}, which is your maximum delay of the register $t_{cq,cd}$ is the contamination of the minimum delay of the register. So this is max delay, this is min delay, this t setup and hold is clock constraint and then t_{logic} is basically your maximum delay of combinational logical block and $t_{logic,cd}$ is basically your delay because of the logic itself.

So under ideal conditions as I told to you in our earlier discussion that t_{clk1} will be exactly equals to t_{clk2} and the worst-case propagation determine the minimum clock period require for the sequential circuit is this much. I think you will understand, see t_{cq} plus t_{logic} plus t_{setup}, why it is like that? Because your minimum clock period of the input clock must take into account the setup time violations or setup time logic t_{logic} is basically the contamination delay of the combinational logical block or at this stage is the total delay of the combinational logical block and t_{cq} is the maximum delay of the register.

Because you see the signal passing through registers through combination logical blocks and then finally going to the output. So the minimum capital T, capital T means the clock period, right? This is the clock period should be at least greater than t_{su}, right? Because otherwise you will have a problem tlogic also the same thing and t_{cq} also the same thing, so capital T should be greater than t_{cq} plus t_{logic} plus t_{su}. What is t_{hold}? t_{hold} is that I discussed with you.

After the rising edge of the clock the time till which the data should remain stable is defined as t_{hold}. And it is given as $t_{logic,cd}$ minus…

(Refer Slide Time: 15:04)



Synchronous Timing Basics

A positive edge triggered system is assumed. Where –

$t_{c\_q,cd}$ = contamination or minimum delay of the register

t_{c_q} = maximum delay of the register

t_{su} = Setup time

t_{hold} = hold time

$t_{logic,cd}$ = Contamination delay of combinational block

t_{logic} = Maximum delay of the combinational block

t_{clk_1}, t_{clk_2} = Positions of the rising edges of the clocks CLK_1 and CLK_2.

Pipelined datapath circuit and timing parameters

Under ideal conditions ($t_{clk1} = t_{clk2}$), the worst case propagation delays determine the minimum clock period required for this sequential circuit,

$$T > t_{c\_q} + t_{logic} + t_{su}$$

$$t_{hold} < t_{c\_q,cd} + t_{logic,cd}$$

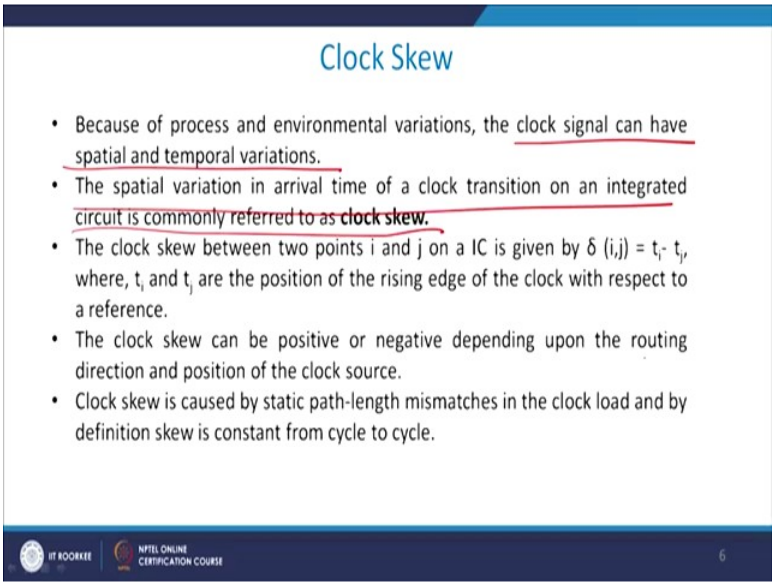
So t_{hold} is equals to $t_{c\text{-}q,cd}$, what is $_{tc\text{-}q,cd}$? It is the contamination delay of the register of the minimum delay of the register plus $t_{logic,cd}$ which is contamination delay or the minimum combination and you can understand why? So this logic will have say one millisecond delay, right? And then this is having 1 millisecond delay, right? And let us also assume that this is having 1 millisecond delay.

So capital T should be at least larger than equal to 3 milliseconds. If it is not then any of the 3 operations will have to be violated which is not possible in a sequential logic, right? And the worst-case propagation delay requires that the delay determines the minimum clock period required for the sequential logic. I will give an example. For example if you are in a situation where you are using a sequential delay for a very stress condition.

What does it mean? That your voltage levels and current levels are very high then what does it do? Is that it gives you the worst-case propagation delay for a sequential logical block, right? So this gives you the worst-case preparation delay in a sequential logical block. As you can see capital T is greater than t_{cq} plus $t_{pd\ logic}$ plus t $_{setup}$ plus t $_{logic}$ and t $_{hold}$ should be less than $t_{c\text{-}q,cd}$ and $t_{logic,cd}$.

Now why t_{hold} should be less than that? Let us suppose t_{hold} is greater than that then what will happen is that either of these 2 problem areas will be inserted into the logic principle and there will be an issue as far as output is concerned.

(Refer Slide Time: 16:42)

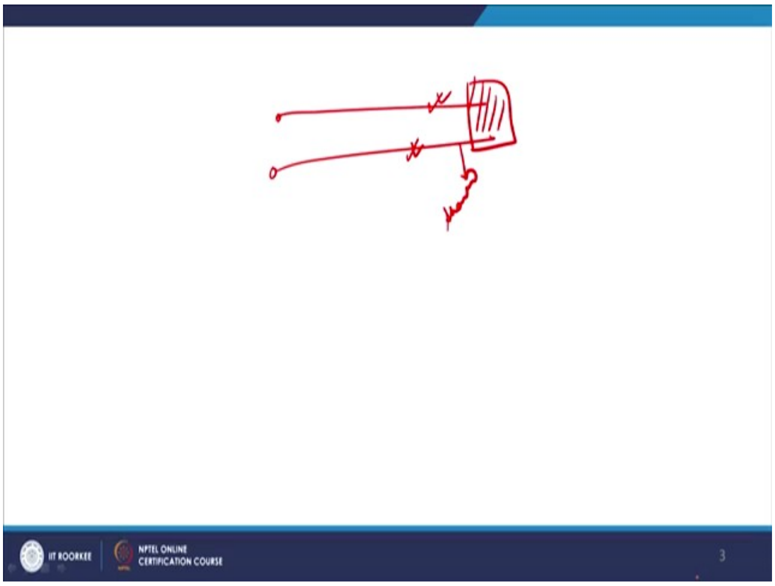


Clock Skew

- Because of process and environmental variations, the clock signal can have spatial and temporal variations.
- The spatial variation in arrival time of a clock transition on an integrated circuit is commonly referred to as **clock skew.**
- The clock skew between two points i and j on a IC is given by δ (i,j) = t_i - t_j, where, t_i and t_j are the position of the rising edge of the clock with respect to a reference.
- The clock skew can be positive or negative depending upon the routing direction and position of the clock source.
- Clock skew is caused by static path-length mismatches in the clock load and by definition skew is constant from cycle to cycle.

So let us look at now at clock skew. We have understood the basic idea about the delay concepts, let us come to clock skew because of process variations and environmental variations the clock signals can have spatial and temporal variations, so how did you define clock skew? The spatial variation in the arrival time of the clock on an IC is referred to as clock skew.

(Refer Slide Time: 17:10)

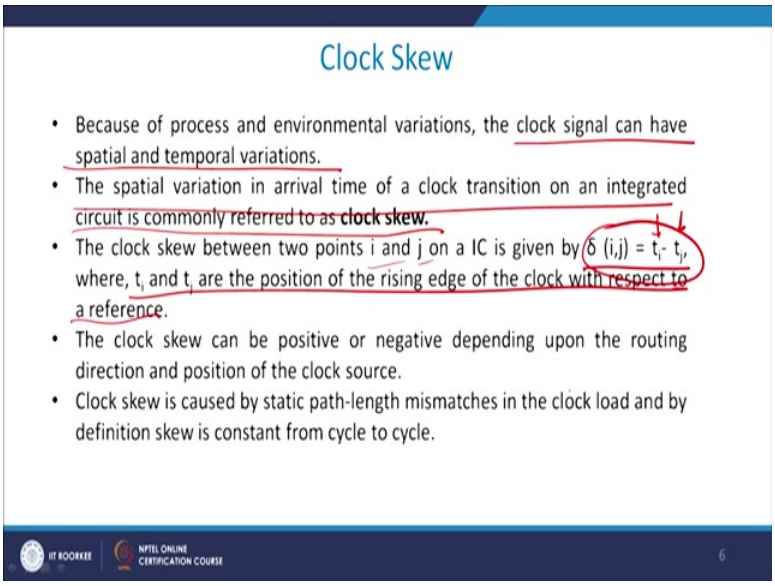


Which means that let me give you an example, let us say I have a IC here and this is driven by 2 external clocks. If this clock take higher time as compared to this for whatever reasons we define that this clock is basically having a skew design and you need to formulate it, why these are required? Because see until and unless you send all the signals in transparent mode as well as properly you will not be able to evaluate the output signal.

And that is what is happening here, so what is the clock skew? The spatial variation in the arrival time of a clock transition or an IC is referred to as clock skew. Now the clock skew between the 2 point i and j is given by δ (i, j) but t_i minus t_j and this is quite important also. So for example if you have clock here.

Maybe I can, right? Maybe I can discuss here, right? But nonetheless, let me see how it works out.

(Refer Slide Time: 18:20)



Clock Skew

- Because of process and environmental variations, the clock signal can have spatial and temporal variations.
- The spatial variation in arrival time of a clock transition on an integrated circuit is commonly referred to as **clock skew.**
- The clock skew between two points i and j on a IC is given by $\delta\ (i,j) = t_i - t_j$, where, t_i and t_j are the position of the rising edge of the clock with respect to a reference.
- The clock skew can be positive or negative depending upon the routing direction and position of the clock source.
- Clock skew is caused by static path-length mismatches in the clock load and by definition skew is constant from cycle to cycle.

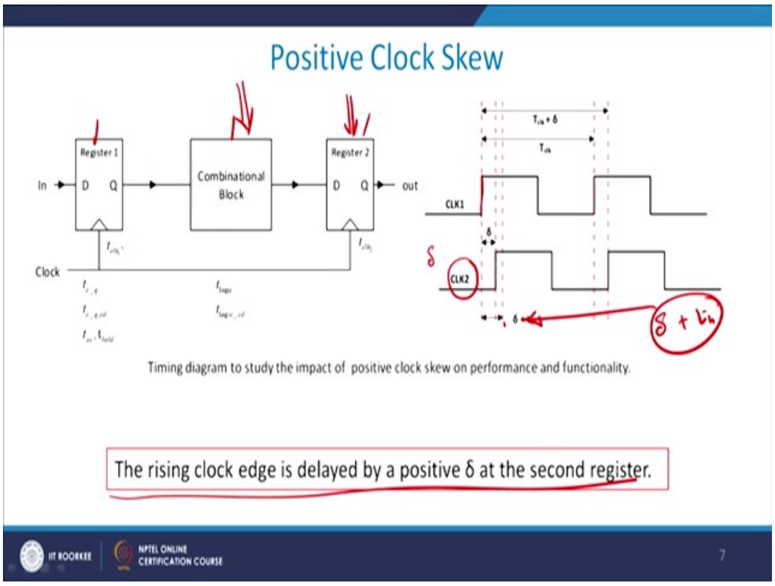
Now you see that the spatial variation in time of arrival of the clock. Transition on an IC is referred to as a clock skew. In the arrival time, please understand it is not on the terms of delay, it is not in the terms of frequency it is arrival time it is important. So if it has arrived later than the clock skew it is known as positive skew. If it arrives ahead of the rising edge of the clock we define that to be as the premature case.

So one has to be very careful what is clock skew. As I discussed with you the clock skew between 2 points in an IC with having i and j as the subscript, we can write δ(i, j) is equal to t_i minus t_j where t_i is the time taken for the data to reach at the particular point and t_j is basically the output point with respect to the rising edge. So t_i and t_j are the position of the rising edge of the clock with respect to a reference.

Now the clock skew can be positive or negative depending upon the routing of direction. So I will like you to find it out. If the direction of the clock and the data is same you will get positive skew, if the direction of the clock and the direction of the data moment is opposite to each other we will get a negative skew and I would like you to find out why so? Now clock skew will remain same from cycle to cycle, right? It does not change.

Because it does not depend upon high mismatches and so on and so forth. It only depends upon clock load but unlike jitter which we will see later on clock skew is very stable in terms of the output which is there with me, right? The clock skew as I discussed with you positive or negative depending upon the routing and the direction of the positive nature of the clock source.

(Refer Slide Time: 20:15)



Positive Clock Skew

Timing diagram to study the impact of positive clock skew on performance and functionality.

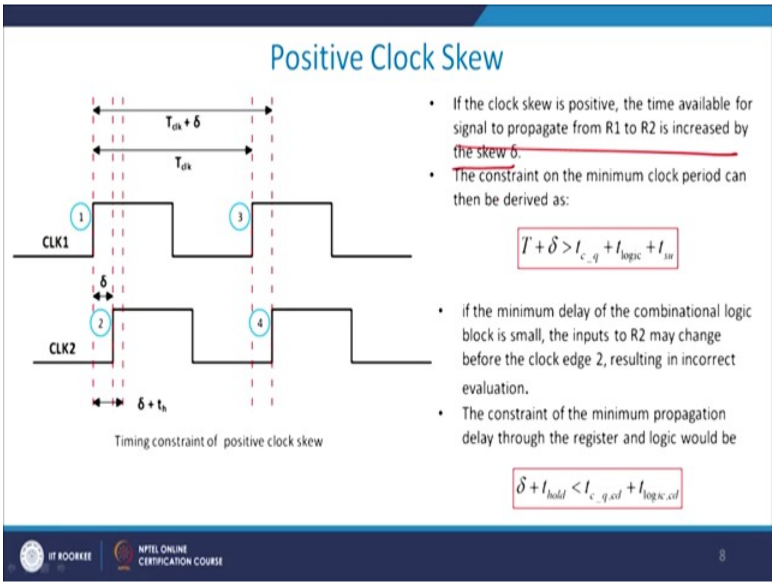
The rising clock edge is delayed by a positive δ at the second register.

So let us look at what is known as a positive clock skew. What is a positive clock skew is, I have register 1, register the combinational logic and register 2 is there with me. Now I have a tclk1 tclk2. So at the rising edge of the clock 1 nothing is happening because in clock 2 you already have a skew which means that the rising edge will shift by right to accept this data and therefore you get δ is equal to this much.

Now in reality it has to also hold a data equals to t_h, the total delay is basically δ plus t_h and that is what we have seen at this particular point, fine. Is it conceptually cleared? Right. So combination logical blocks job is to accept the data and then do some logical operation on it and then send the data to the registered number 2. So till a time this happens register number 2 is sitting ideal.

And what does combinational logical block try to do? It tries to accept the data from register 1 and processes it and give it to register 2. Register 2 within a few period of time manipulates it and send it to the output, right? So the rising edge is delayed by a positive δ the second register, you got the point. Why? Because if you see carefully by the time this reaches the second register the clock one might have actually gone behind as compared to clock 2 or maybe ahead also and therefore you can get a rising edge delay in terms of Δ in second register.

(Refer Slide Time: 22:02)



Positive Clock Skew

Timing constraint of positive clock skew

- If the clock skew is positive, the time available for signal to propagate from R1 to R2 is increased by the skew δ.
- The constraint on the minimum clock period can then be derived as:

$$\boxed{T + \delta > t_{c\_q} + t_{logic} + t_{su}}$$

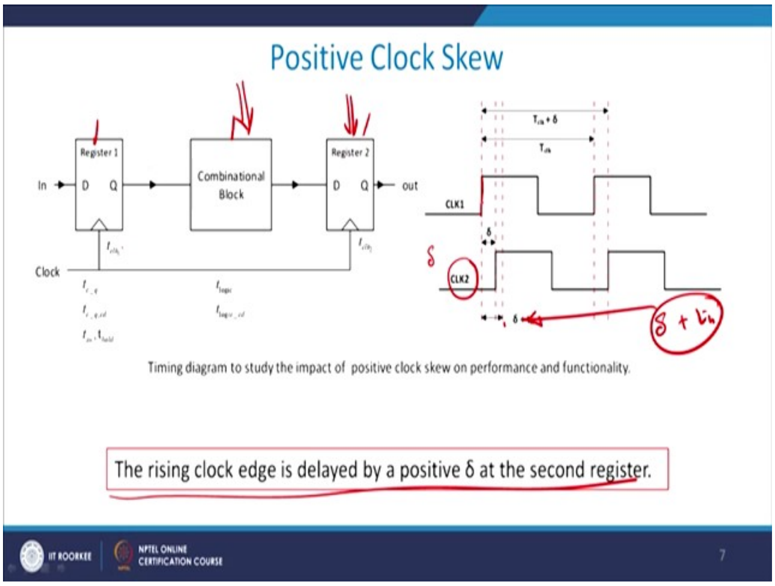
- if the minimum delay of the combinational logic block is small, the inputs to R2 may change before the clock edge 2, resulting in incorrect evaluation.
- The constraint of the minimum propagation delay through the register and logic would be

$$\boxed{\delta + t_{hold} < t_{c\_q,cd} + t_{logic,cd}}$$

8

Okay, as I discussed if the clock skew is positive the time available for signal to propagate from R_1 to R_2 and increased by δ, I suppose this is clear no need to explain.

(Refer Slide Time: 22:13)



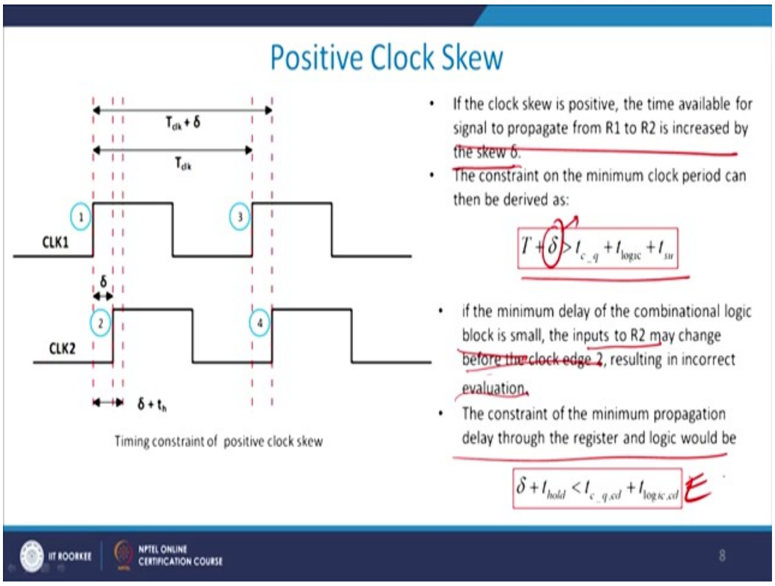
Positive Clock Skew

Timing diagram to study the impact of positive clock skew on performance and functionality.

The rising clock edge is delayed by a positive δ at the second register.

For example if the clock skew is positive then I get a larger time to integrate my design, right? And therefore it is much easier to do that.

(Refer Slide Time: 22:27)



Positive Clock Skew

Timing constraint of positive clock skew

- If the clock skew is positive, the time available for signal to propagate from R1 to R2 is increased by the skew δ.
- The constraint on the minimum clock period can then be derived as:

$$\boxed{T + \delta > t_{c\_q} + t_{logic} + t_{su}}$$

- if the minimum delay of the combinational logic block is small, the inputs to R2 may change before the clock edge 2, resulting in incorrect evaluation.
- The constraint of the minimum propagation delay through the register and logic would be

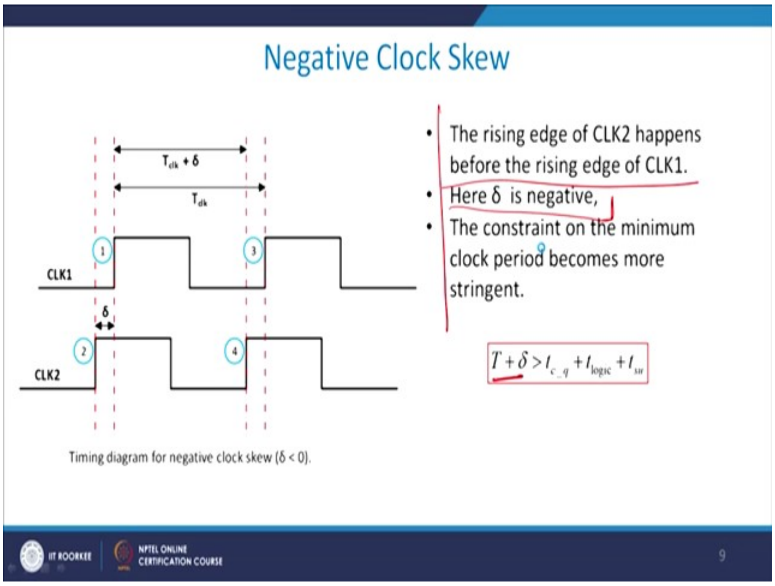
$$\boxed{\delta + t_{hold} < t_{c\_q,cd} + t_{logic,cd}}$$

And therefore if you look very carefully positive clock skew is always beneficial in nature. Now let us look at the constraint of the total timing delay and the total timing delay is given by T plus δ, why T plus δ? Because that is the capital T is the total time period when the system is going from 0 to 1 and 1 to 0 and the value of δ can be found out from this relationship.

But the value of δ will not be varying, will be fixed one for the fixed cases of interest. So if the minimum delay of the combination logical block is small the input register 2 may change before the clock edge of 2 resulting in incorrect evaluation, you got a point. That means if the combination logical block has very low delay then the output of gate number 3 exactly matches with the gate number 1 rising pulse.

But that has to be available to gate number 2 in order to evaluate that person, right? But if it is not available, it is lost. So that is the problem area of T logic or transistor logic. The constraint of minimum propagation delay through the register in the logic would be as I discussed with you, it is not there would be this much, right? So the constraint of minimum propagation delay, what is the minimum propagation delay? It is the maximum error will be given by this particular formula here, right? That is what we get from the positive clock skew.

(Refer Slide Time: 23:57)

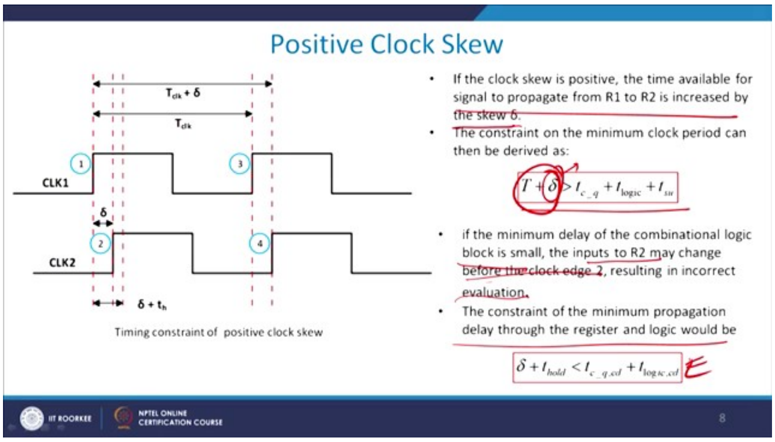


Negative Clock Skew

Timing diagram for negative clock skew (δ < 0).

- The rising edge of CLK2 happens before the rising edge of CLK1.
- Here δ is negative,
- The constraint on the minimum clock period becomes more stringent.

$$T + \delta > t_{c\_q} + t_{logic} + t_{su}$$



Positive Clock Skew

Timing constraint of positive clock skew

- If the clock skew is positive, the time available for signal to propagate from R1 to R2 is increased by the skew δ.
- The constraint on the minimum clock period can then be derived as:

$$T + \delta > t_{c\_q} + t_{logic} + t_{su}$$

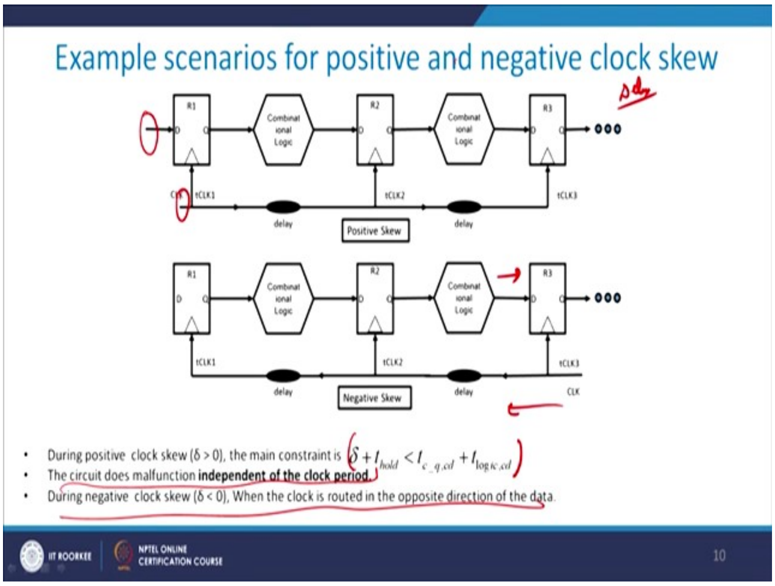
- if the minimum delay of the combinational logic block is small, the inputs to R2 may change before the clock edge 2, resulting in incorrect evaluation.
- The constraint of the minimum propagation delay through the register and logic would be

$$\delta + t_{hold} < t_{c\_q,cd} + t_{logic,cd}$$ E

Let me come to negative clock skew. Negative clock skew is just the reverse of positive. In the positive clock skew the data part and the person who is looking at the skew they were in the same direction. So if the data path is going from X to Y the clock was also moving from X to Y. Here the reverse is happening when the clock is moving from left to right the device is operating from right to left and therefore we define negative is to be negative in nature.

The rising edge of the clock 2 happens before the rising edge of clock 1 and therefore δ is negative and therefore here δ is kept to be negative. The constraint on the minimum time period becomes more stringent that T plus δ should be greater than or equal to this much. Why it is δ? Because this δ is a minimum difference between the set up time and the rising edge of the pulse for a MOS device in the study and that the reason it is converted into this simple form where you can understand the T plus δ which is basically your this quantity. It should be always greater than this thing t_{su}, t_{cq} and t_{logic}.

(Refer Slide Time: 25:18)



Example scenarios for positive and negative clock skew

- During positive clock skew (δ > 0), the main constraint is $(\delta + t_{hold} < t_{c\_q,cd} + t_{logic,cd})$
- The circuit does malfunction **independent of the clock period.**
- During negative clock skew (δ < 0), When the clock is routed in the opposite direction of the data.

IIT ROORKEE | NPTEL ONLINE CERTIFICATION COURSE 10

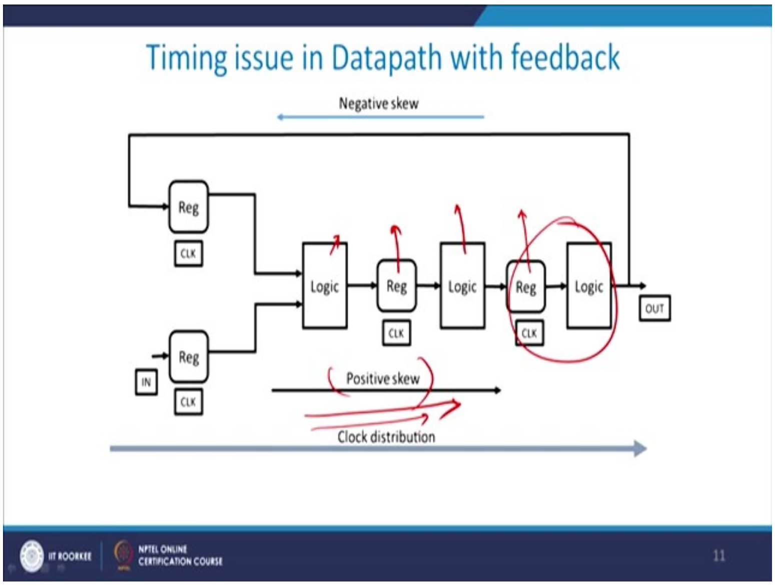
With this we come to an example scenario of a positive and negative clock skew, right? And I can say here that for positive clock skew this input coming from R_1 is sort of a delay element which goes to R_3 directly so R_1 to R_3, so there are 3 registers R_1, R_2 and R_3 and there are 2 combinational logical blocks, combinational logic 1, combinational logic 2 but please understand to move from R_1 to R_3 it has to be properly tuned in terms of potential, so that it can move easily from point A to point B This is the case of positive skew.

Positive skew means the data train which is this one and the clock which is this one they move exactly in the same direction, they move exactly in the same direction. So the delay of the combinational logical block is basically your internal delay. Similarly if the clock is in the opposite direction as compared to data you will have a negative delay at this point and this point.

So during the positive edge of the clock δ is greater than 0 and therefore I should maintain δ plus t_{hold} should be less than equal to this quantity. And the circuit does malfunction independent of the clock period. Sometimes there is a malfunctioning in the clock and therefore it stops working. During negative clock skew when the data is routed to the opposite direction of the clock we get the negative skew in this case.

So we understood what is the positive skew, what is a negative skew and it works.

(Refer Slide Time: 26:54)

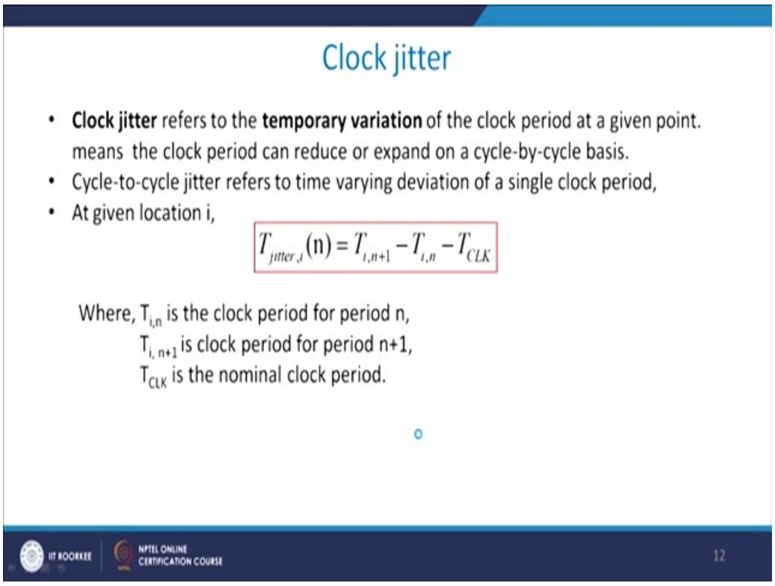


Timing issue in Datapath with feedback

Now timing issue in a data path with feedback. Let us suppose I have a negative skew and this is a feedback which is there with my… have a logic here. I have logic which takes care of the logical experts then we have got a register in our house then this is basically a logic analyzer. Same is the case with of this one, these 3 are basically simple players in this game. So this is basically a positive skew.

Because this is by data train path and this is my moment of the various blocks from left to right and by using a simple rather than a problem area.

(Refer Slide Time: 27:36)



Clock jitter

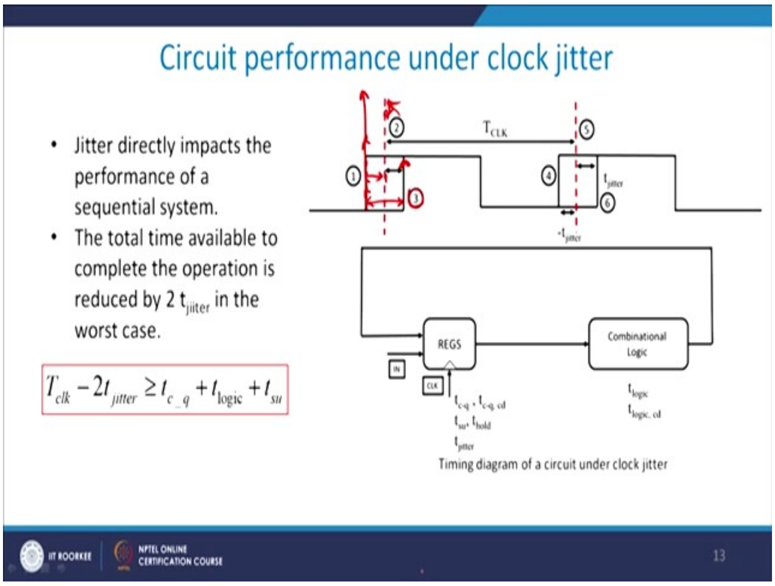
- **Clock jitter** refers to the **temporary variation** of the clock period at a given point. means the clock period can reduce or expand on a cycle-by-cycle basis.
- Cycle-to-cycle jitter refers to time varying deviation of a single clock period,
- At given location i,

$$\boxed{T_{jitter,i}(n) = T_{i,n+1} - T_{i,n} - T_{CLK}}$$

Where, $T_{i,n}$ is the clock period for period n,
$T_{i,\, n+1}$ is clock period for period n+1,
T_{CLK} is the nominal clock period.

Let me come to one more important point and that is clock jitter and we will take up this one in the detail manner in the next turn. Clock jitter refers to temporary variation of the clock period. So the clock skew was spatial variation whether there is a physical movement of the clock from point A to point B.

(Refer Slide Time: 27:59)



Circuit performance under clock jitter

- Jitter directly impacts the performance of a sequential system.
- The total time available to complete the operation is reduced by 2 t_{jiiter} in the worst case.

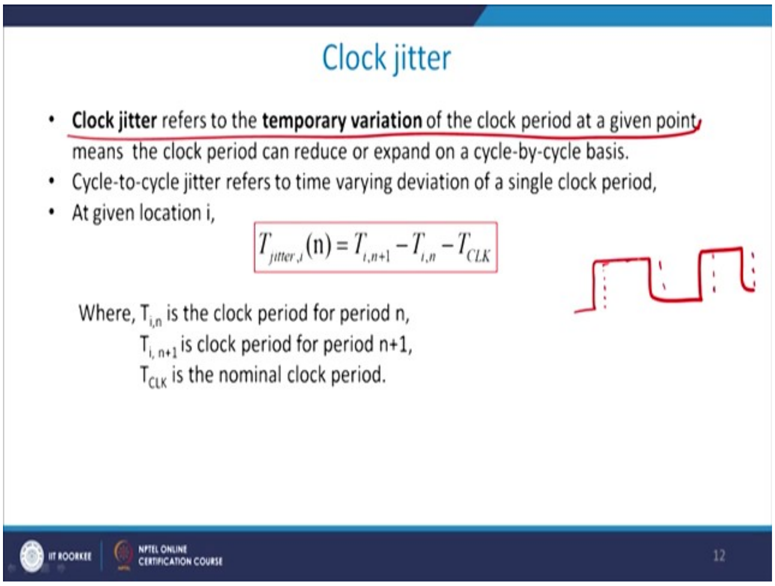
$$T_{clk} - 2t_{jitter} \geq t_{c\_q} + t_{logic} + t_{su}$$

Timing diagram of a circuit under clock jitter

Temporary variation primarily means that you do have this much amount of variations so you initially had 2. At 2 you had this thing and then what happened is after part 2, at one suppose you are giving a positive edge to get design here. But what has happened is that you have to wait till this edge number 3 to come for which you need to wait till this much amount of time domain when the next clock higher edge comes.

So initially it was triggered at 1, it was supposed to be triggered at 1, what has happened is, it is not triggered to either at 3 or 2, so there will be a small this t_{jitter} which will be there. This is known as a temporal variation, in time domain you will see a shift and it can vary from 1 block to another block, the skew remains constant from block to block but the jitter may vary from 1 block to another.

(Refer Slide Time: 29:05)



Clock jitter

- **Clock jitter** refers to the **temporary variation** of the clock period at a given point, means the clock period can reduce or expand on a cycle-by-cycle basis.
- Cycle-to-cycle jitter refers to time varying deviation of a single clock period,
- At given location i,

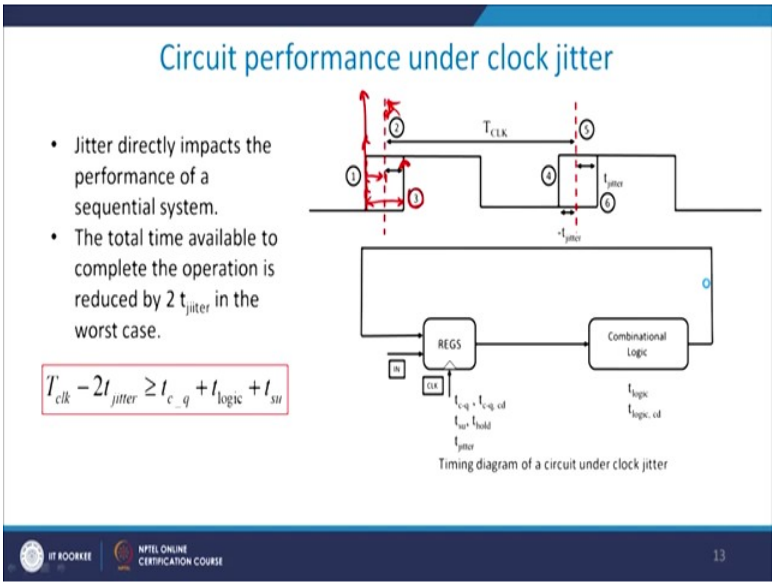
$$\boxed{T_{jitter,i}(\mathrm{n}) = T_{i,n+1} - T_{i,n} - T_{CLK}}$$

Where, $T_{i,n}$ is the clock period for period n,
$T_{i,\,n+1}$ is clock period for period n+1,
T_{CLK} is the nominal clock period.

We will work out the details of it in the later stages but primarily if you look very carefully this case you can see that therefore clock jitter refers to the temporary variation of the clock period at the given period of time. So initially you had this into the consideration. Now what has happened is because of some problem we will discuss that later on that it has not shifted from this side to this side.

Everything has shifted by a particular phase from this side to this side, this shifting is there.

(Refer Slide Time: 29:31)



Circuit performance under clock jitter

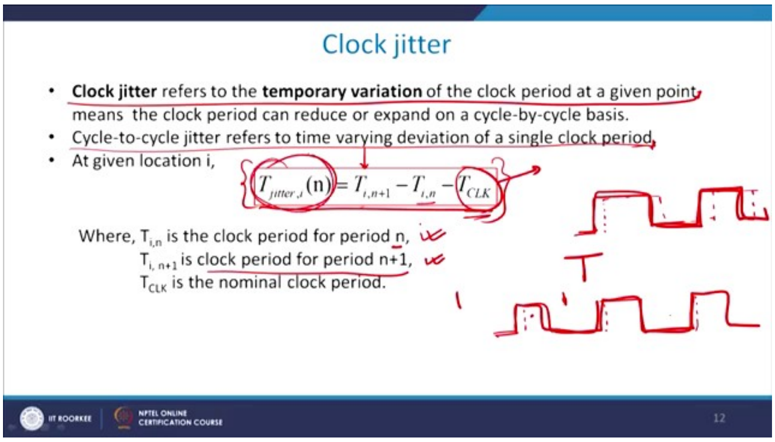
- Jitter directly impacts the performance of a sequential system.
- The total time available to complete the operation is reduced by 2 t_{jiiter} in the worst case.

$$\boxed{T_{clk} - 2t_{jitter} \geq t_{c\_q} + t_{logic} + t_{su}}$$

Timing diagram of a circuit under clock jitter

As you can see this shift is there. So this was initially here this shifted here.

(Refer Slide Time: 29:35)



Clock jitter

- **Clock jitter** refers to the **temporary variation** of the clock period at a given point, means the clock period can reduce or expand on a cycle-by-cycle basis.
- Cycle-to-cycle jitter refers to time varying deviation of a single clock period.
- At given location i,

$$T_{jitter,i}(n) = T_{i,n+1} - T_{i,n} - T_{CLK}$$

Where, $T_{i,n}$ is the clock period for period n,
$T_{i,\ n+1}$ is clock period for period n+1,
T_{CLK} is the nominal clock period.

And as you can see therefore there is a cycle to cycle refers to time varying deviation of a single clock period. So you had a single clock period of capital T. Now because of cycle to cycle variation in the jitter your new value at location I will be given as $T_{i\ jitter}$ to be equals to $T_{i(n+1)}$ minus T_{in} minus T_{clk}, where T_{in} is basically the clock period for period n and T_{in} plus 1 is for n^{th} plus 1.

So when you vary from first-period to the second period, so this is one and then when you vary for the second period there is a lateral shifting. So what has happened is that we defined t_{jitter} to be this minus this the difference because that is extra 1 minus clock, why? Because clock is common to both, so when you subtract that the difference will come out to you between the various values of skew and jitter right.

So we will take-up the next part of the strategy in the next clocking strategy in the next morning. Thank you very much.

(30:37)

###

???

(31:04)

Okay so clock jitter refers to temporary variation of the clock period at a given point of time which means that the clock can expand or reduce from one cycle to another. So this is one cycle, let us suppose this is one cycle and I have another cycle, so this is one cycle, this is another one cycle and then this is another cycle. So between this cycle and this cycle you might have a difference in the rising edge of the clock this is known as jitter.

So this is a temporary domain whereas skew is special domain. Skew does not change from cycle to cycle whereas jitter changes from cycle to cycle. We define jitter at any particular point in given location i to be equals to $T_{i,\ n+1}$ which is basically the clock period for $(n + 1_{)}^{th}$ minus Ti n minus at nth clock. So when you go from $T_{(n+1)th}$ clock to n^{th} clock the difference between the 2 is basically your jitter.

But you understand within that you already have the clock period common to both of them. So if use subtract, when you subtract that you actually get the overall jitter which is there, right? So this is basically from cycle to cycle. Once you have one cycle then another cycle the jitter varies from cycle to cycle, right? With this we have finished with this first module of your sequential logical clocking strategies.

In the next block we will take care of the combined effect of skew and jitter on the performance of a sequential logical block, thank you very much.

.